ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

Adaptación de un entorno inteligente mediante balizas BLE para invidentes

MEMORIA

Autor: Guillermo García de la Fuente

Tutor: Ignacio R. Matías Maestro

Pamplona, julio 2014

AGRADECIMIENTOS

A todos aquellos sin cuyo apoyo esto no habría sido posible:

- A mis padres
- A mi pareja
- A mi tutor
- A los profesores de Ing. de Telecomunicación de la UPNA
- A Mikel Meoki y Simón Santestéban de Embeblue
- A todos mis compañeros de carrera

Resumen

En este documento se va a exponer un Proyecto de Fin de Carrera guiado a demostrar cómo mediante el diseño de Entornos Inteligentes que interactúen con el usuario respondiendo a sus necesidades pueden ser adaptados espacios que hasta hoy no permitían un cómodo acceso a ciertas personas.

Concretamente se tratará de demostrar cómo con un presupuesto relativamente bajo se puede crear una aplicación básica para Android que localice al usuario en un espacio interior, donde las tecnologías como GPS no pueden ayudarnos, y le guíe hasta los objetivos que él mismo haya designado.

La localización de un usuario es una demanda que ya comparten numerosas aplicaciones móviles para servicios como mapas, localización establecimientos cercanos, mostrar una publicidad selectiva... y todo ello ha derivado a que se requiera incluso una localización en interiores. Dicho campo se encuentra muy en auge con las nuevas apariciones de balizas y es por ello que se decidió orientar su uso a ayudar a aquellos que más lo necesiten.

ÍNDICE

Contenido

1. Introducción

La necesidad por saber dónde nos encontramos ha estado siempre presente en nuestro mundo. No obstante, si ya era algo útil de por sí el tener conocimiento acerca de la propia posición y elementos cercanos hoy en día la tecnología nos permite interactuar con todo ello de una manera mucho más interesante y efectiva. Desde localizar restaurantes cercanos hasta conocer alternativas desde la propia posición a una ruta de tráfico con alta densidad en el mismo momento.

Todo ello con el tiempo ha derivado en servicios más concretos de localización hasta que hemos llegado a la localización en interiores. Este área se ha convertido en uno de los campos más prometedores actualmente haciendo que los servicios de información y comunicación basados en la posición de individuos u objetos se estén convirtiendo en nuevos modelos de negocio capaces de revolucionar los modos de rentabilizar estructuras.

Dentro de todo ello podemos encontrar el llamado "context aware". Este crea relación entre los cambios de contexto de los usuarios, sean estos su posición, recursos cercanos, etc. dando la posibilidad de facilitar la interacción del usuario con distintos elementos, personalizar como ha de reaccionar su entorno, la información que ha de recibir...

Por otro lado nos encontramos ante un importante sector del público, los invidentes, que podrían beneficiarse ampliamente de los servicios que un sistema de localización en interiores con las propiedades antes citadas. Se encuentran con graves problemas a la hora de realizar algunas tareas rutinarias como la compra en el caso de que se hayan vuelto a redistribuir los productos en las estanterías colocándolos en nuevas posiciones.

Esto nos lleva a que actualmente aún no encontramos una solución adecuada y adaptada para la localización en espacios donde el GPS y otros sistemas de posicionamiento exterior no llegan y que sirva para este público concreto.

Además se añaden problemáticas de carácter técnico y económico. Técnicas en el sentido de que la localización en interiores se enfrenta a problemas muy superiores a los de la localización en espacios abiertos y económicas porque generalmente requieren una alta inversión en infraestructuras fijas (Balizas, puntos de control, sensores...).

El sistema que se va a proponer se basa en la recepción de la señal procedente de unas balizas BTLE (Bluetooth Low Energy) y de los puntos de acceso Wi-Fi. Mediante ellos se localizará en un mapa previamente facilitado y con unos puntos a recorrer especificados por el usuario para al final guiarle a través de ellos.

1.1 Motivación

Para una persona invidente o con un grado muy alto de ceguera es todo un reto poder realizar la compra por sí mismo. Especialmente, si como se ha mencionado antes, los supermercados cambian los productos de sitio. Puede que no ocurra de forma regular, pero es cuando menos algo que obliga prácticamente a que una persona de estas características necesite a alguien que le acompañe, guíe o ayude a realizar una tarea cotidiana para todos.

Este PFC pretende ayudar a esas personas otorgándolas una mayor independencia y libertad de movimiento.

1.2 Objetivos

Este PFC tiene como objetivo el diseño, implementación y comprobación de la viabilidad de un prototipo de sistema de posicionamiento en interiores basado en redes

Wi-Fi pero especialmente en señales BTLE emitidas por balizas de bajo coste distribuidas estratégicamente por una determinada superficie. Los puntos principales a perseguir son los siguientes:

- Diseño de una aplicación en Android cuyo uso sea sencillo para personas con discapacidad visual.
- Dicha aplicación ha de ser capaz de almacenar varias listas de la compra con elementos a localizar en la superficie objetivo.
- Además debe ordenar los elementos introducidos para conseguir una ruta más o menos eficiente.
- Una vez obtenida la lista y ejecutada la aplicación esta ha de ser capaz de dar indicaciones al usuario que le guíen a través de la superficie hasta donde se encuentren los productos escogidos.

2. Estado del Arte

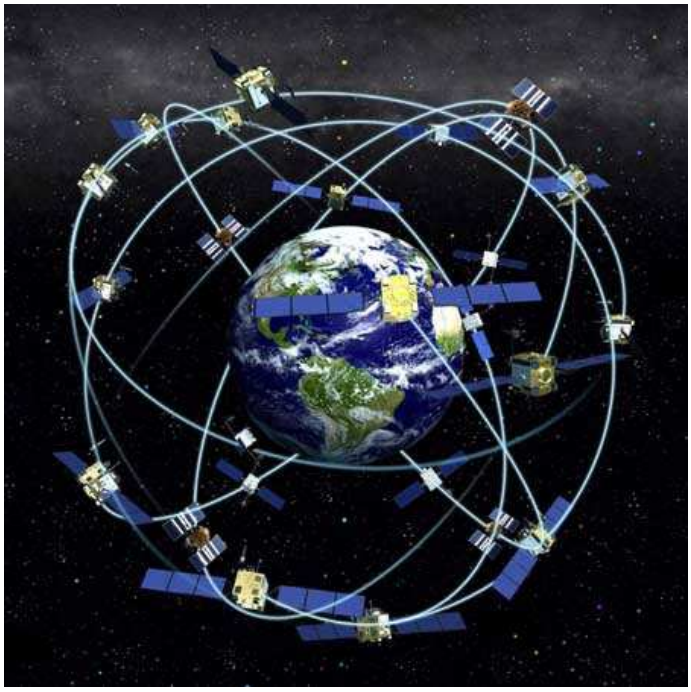
En la actualidad gran parte de la población posee un ordenador personal, portátil, un smartphone... y es raro que no vengan ya preparados con tecnologías inalámbricas. No obstante, todas esas tecnologías cuya función principal es la de transmisión de datos pueden ser aprovechadas para otros usos como es el que se pretende en este proyecto.

A continuación se enumerarán algunas de las tecnologías y sistemas que se pueden encontrar.

2.1 Tecnologías de localización

2.1.1 GPS

El GPS es por excelencia el método de posicionamiento más conocido. Al igual que Galileo se basa en la recepción de la señal de una constelación de satélites que orbitan alrededor del planeta emitiendo para todo el que pueda recibirles.



Cuando el dispositivo recibe al menos la señal de 4 satélites, acompañada del instante en que se enviaron y de la posición del satélite, es capaz de triangular su posición sobre la superficie de la tierra con una cierta precisión que será mayor cuantos más satélites sea capaz de recibir.

Este sistema se encuentra con el problema de que depende de encontrarse en línea de visión directa con los satélites, lo que le trae problemas en determinados espacios como ciudades con rascacielos. Otro punto a añadir es que dada la distancia desde la que se emiten la señales la señal llega al receptor con muy poca potencia. Esto hace que en el interior de edificios pierda casi por completo su utilidad.

2.1.2 IEEE 802.11

Protocolo de comunicación típico de las W-LAN. Varios de los protocolos contenidos en él emiten en 2.4 GHz y otros en 5 GHz.

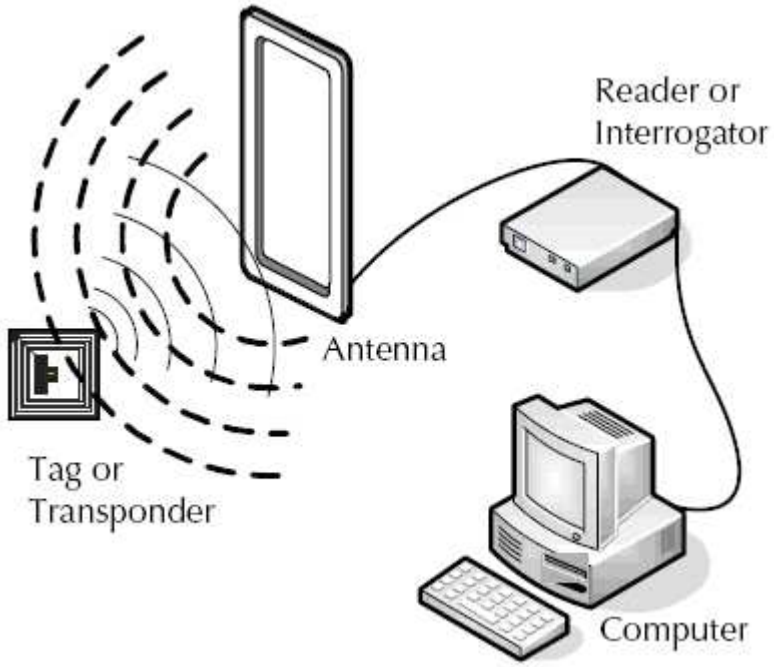
Su señal atraviesa los muros con cierta dificultad, por lo que no es habitual que se usen para algo distinto que crear una red local. Además su señal también se ve muy afectada por otros factores como obstáculos móviles que absorben parte de su energía, reflexiones, difracción y dispersión. Con lo que además acabamos con el problema del multitrayecto.

Pese a todo las señales recibidas pueden aprovecharse como se verá más adelante y es un sistema que no requiere hardware adicional pues todos los teléfonos actuales tienen dicha tecnología incorporada.

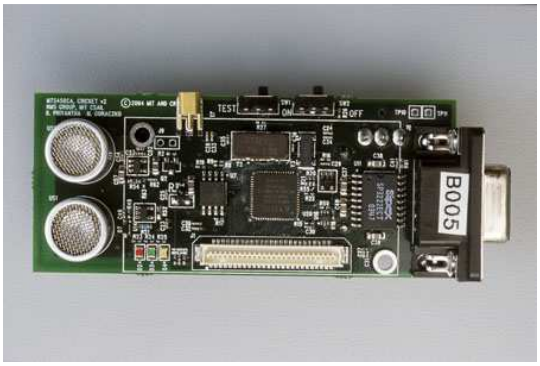
2.1.3 RFID

El sistema RFID se basa en etiquetas de radiofrecuencia. Estas poseen una antena que al ser excitada por un transmisor emite señal en el caso de las pasivas o, en el caso de las activas, que emite a menudo.

De este modo un usuario que pretenda obtener su posición habrá de utilizar un lector para dichas etiquetas y con la información obtenida de ellas podrá posicionarse.



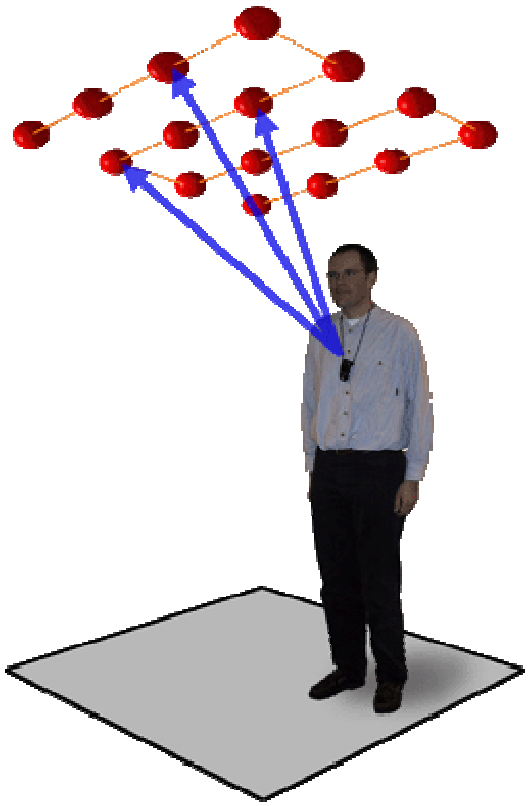
Un ejemplo es el sistema Cricket, diseñado por ingenieros de MIT, que combinando la señal RFID con otra emitida por ultrasonidos es capaz de dar una precisión de 2 cm.



Aunque en un principio podría no parecer viable la localización mediante RFID al ser necesario un lector apropiado, se está extendiendo que los conocidos como "Smartphones" posean la tecnología NFC, que es suficientemente similar e incluso puede simular tags de RFID. Es por ello que quizás puedan ser utilizadas también etiquetas RFID para localizarse con dichos dispositivos sin aumentar el coste debido a tener que adquirir lectores.

2.1.4 Sistema BAT

Este sistema difiere del resto en que se basa en ultrasonidos de banda ancha. El sistema se basa en dispositivos móviles que emiten una señal que es recibida por los distintos receptores colocados en el techo. Mide la distancia entre el emisor y el receptor basándose en el tiempo de llegada y calcula la posición del emisor por multilateración con una precisión de unos 2 cm.



2.1.5 Localización por telefonía móvil

Esta alternativa, que no precisaría de hardware adicional, se basa en la localización por la propia señal del teléfono. Por desgracia su precisión sería muy baja y aunque se nos pueda localizar en una célula determinada de una ciudad el error puede ser de hasta 50 metros, lo que la descarta como opción viable para la localización en interiores.

2.1.6 Bluetooth

El usuario escanearía el área en búsqueda de dispositivos cercanos y mediante los localizados y la potencia recibida podría localizarse como con el protocolo 802.11.

Por desgracia la precisión es aproximadamente de 1.5 metros, aunque no está mal para interiores.

La reciente llegada de la tecnología BTLE, incorporada en Android a partir de su versión 4.3, nos permite la creación de balizas cuya batería pueda tener una autonomía más que suficiente como para colocarlas en puntos fijos y no tener que preocuparnos de ellas en mucho tiempo. Podríamos considerar a la tecnología BTLE como la sucesora de Zig-Bee, otro estándar para la transmisión de datos en la misma frecuencia que utiliza una red de tipo malla y se centra en un bajo consumo.

Otra ventaja es que todos los teléfonos vienen también con dicha tecnología disponible, por lo que tan sólo es requisito que su sistema Android esté debidamente actualizado.

Actualmente podemos encontrar ya varios productos en el mercado que se basan en BTLE para tener siempre localizados ciertos objetos mediante unos "Stickers" con BTLE y una aplicación para Android que los detecte.

2.2 Algoritmos y técnicas de localización

De las tecnologías anteriores al final se escogieron para realizar este Proyecto tanto Bluetooth LE como el protocolo 802.11 como apoyo.

Estas tecnologías compartían las siguientes características determinantes:

- No poseen un alcance muy elevado, lo cual nos permitirá discriminar de un mejor modo localizaciones situadas a unas distancias relativamente cortas unas de otras.
- Aunque atraviesan paredes con cierta dificultad no es tanta como para que suponga un problema y puede permitir cubrir un área relativamente grande sin usar demasiadas balizas.

- Ambos estándares son unas tecnologías ampliamente comunes en la mayoría de los dispositivos que la gente lleva consigo, especialmente en los Smartphones. Esto conlleva realmente dos ventajas. La primera es que su disponibilidad es alta y los precios reducidos. La segunda es que no requieren de hardware adicional para funcionar en los dispositivos del usuario.

Para ellas existen varias técnicas que permiten la localización, entre ellas las siguientes:

2.2.1 ToA (Time of Arrival)

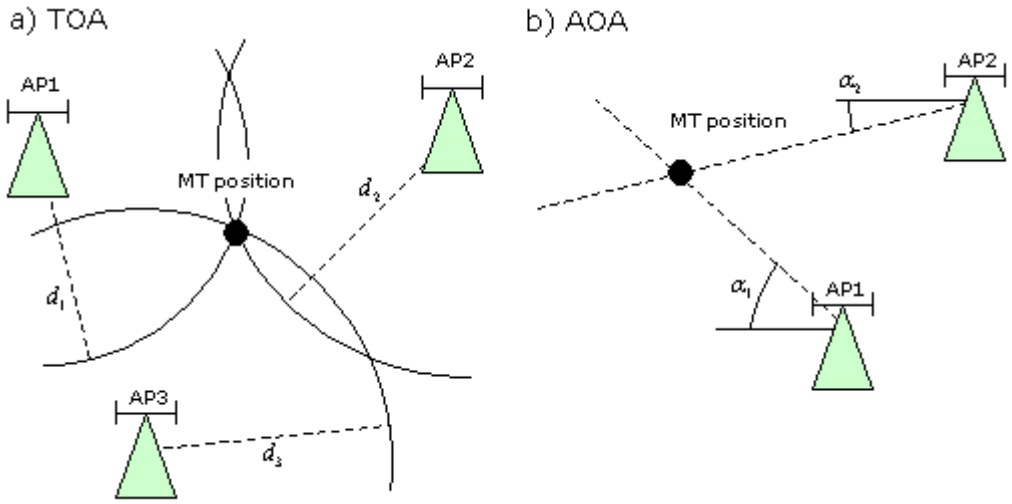
Conocida como "Tiempo de llegada", ésta técnica se basa en el tiempo requerido por una señal electromagnética en recorrer una determinada distancia. Mediante triangulación permite localizar la posición del objeto deseado basándose en la relación lineal que existe entre el tiempo de propagación y la distancia recorrida por la señal.

Por lo general serían necesarios tres puntos de referencia para una localización en dos dimensiones, aunque pueden ser sólo dos puntos en determinadas circunstancias. Por otro lado requiere una gran precisión en el tiempo de referencia y la sincronización de los respectivos relojes.

Podemos encontrar una variación de esta técnica denominada DToA basada en la diferencia entre tiempos de llegada. Esta variación puede usarse tanto combinando dos señales distintas (una electromagnética y otra acústica como en el "Cricket") o una sola señal con varias estaciones base que la reciban (como en el sistema BAT).

Dado que se basa en una medida de tiempo relativa y no en una absoluta puede disminuir el error usando una resolución temporal mayor.

No obstante los problemas derivados por el multitrayecto en interiores pueden ser bastante molestos haciendo que las mediciones de distancia no sean muy precisas. Además un reloj muy preciso sería requisito indispensable.



2.2.2 AOA (Ángulo de llegada)

Esta técnica es sencilla de visualizar pero de compleja implementación. Mediante varias antenas podemos determinar el ángulo incidente de las ondas y deducir su localización basándonos en los parámetros que conocemos como la posición de las antenas y triangulando la posición mediante los ángulos obtenidos.

Podríamos encontrar varios ejemplos de antenas capaces de hacer eso, pero necesitarían la capacidad de poder orientarse para determinar el ángulo del que proviene la señal o constituir un array de antenas cuyo diagrama de radiación pueda modificarse mediante el cambio de fase entre ellas para poder determinar la dirección.

Posee por contra algunos defectos. En el caso de darse multitrayectos podría confundirse el ángulo por el cual llega la señal de interés y llevarnos a localizar el objetivo de un modo erróneo.

Principalmente esta técnica está orientada a espacios despejados o abiertos donde es más complicado que se den multitrayectos.

2.2.3 RSSI (Received Signal Strength Indicator)

La potencia de la señal recibida puede ser accedida fácilmente como parámetro en los sistemas elegidos (BTLE y 802.11) y puede permitirnos conseguir una localización más o menos fiable mediante algunos métodos.

Por desgracia tiene algunos problemas como son obviamente los multitrayectos, reflexiones y otras muchas dificultades del entorno, así como su alta susceptibilidad a las variaciones por la atenuación causada tanto por otros objetos como incluso el modo de sujetar el dispositivo.

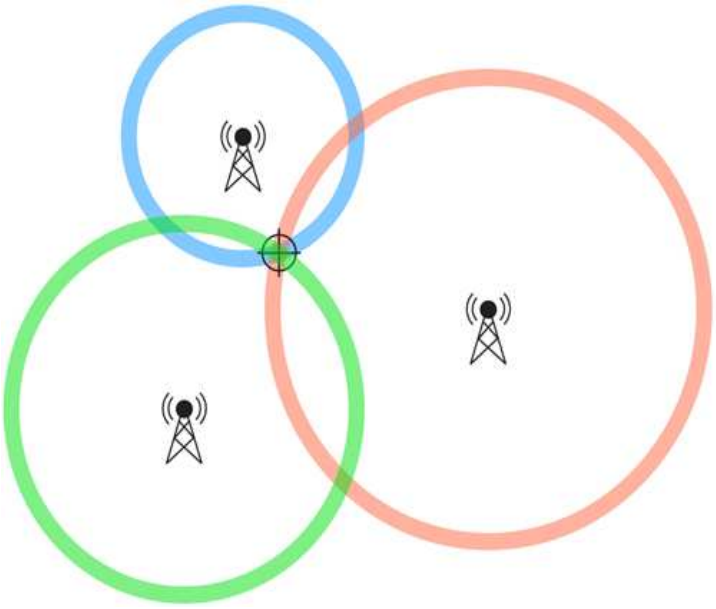
Pese a todo, esta será la opción elegida dado que comparada con las dos alternativas anteriores no requerirá ni de un reloj altamente preciso ni de un hardware complejo para poder llevarse a cabo.

Basándonos en esta ténica encontramos dos opciones a seguir: Triangulación y "Fingerprint" (reconocimiento de patrones)

2.2.3.1 Triangulación

La triangulación es un método muy similar al ya mencinado ToA, pero en lugar de basarse en los tiempos de llegada se basa en la potencia recibida para estimar la distancia hasta el emisor.

Si asumimos que cada una de las balizas emisoras lo hace con la misma potencia y de manera isótropa podríamos basarnos en la ecuación de Friis para determinar nuestra distancia basándonos en la potencia recibida de varias balizas.



Sin embargo nos encontraremos con varios problemas al implementarlo empezando con que las antenas de nuestras balizas no serán ni mucho menos ideales ni nos encontraremos en un entorno que no nos haga recibir señales por multitrayecto o que atenúe la señal por otro lado. Todo ello nos llevará a que aunque pueda ser un buen sistema para exteriores o entornos abiertos, no nos servirá para la localización en interiores.

Por otro lado una ventaja clara hubiese sido que mediante un mínimo trabajo sobre el terreno recogiendo los valores de la posición de las balizas el sistema nos hubiese indicado nuestra posición sin mayor problema con sólo algunos cálculos.

2.2.3.2 Fingerprint (Reconocimiento de patrones)

La técnica conocida como "Fingerprint" o reconocimiento de patrón se basa en la realización previa de un mapa del lugar de trabajo sobre el que se asignará a cada "cuadrícula" o sector un vector con las potencias recibidas por parte de las balizas. Después mediante alguno de los algoritmos existentes se trata de aproximar en cuál de esos puntos se encuentra el usuario.

Dicho mapeado puede realizarse de forma teórica mediante simulación o midiendo punto a punto la potencia recibida y almacenando los datos.

Este método difiere del método de triangulación principalmente en que no se puede obtener una localización que no haya sido definida con anterioridad. Además las posiciones pasen a ser discretas y claramente definidas por quien ha realizado el mapa en lugar de poder ser cualquier punto del espacio estudiado. Cuanto más se divida el espacio estudiado, mayores puntos de precisión se podrán obtener pese al esfuerzo extra que ello implica.

En comparación los cálculos realizados para llevar a cabo la localización mediante reconocimiento de patrones son más sencillos que los que conlleva la triangulación. Estos cálculos son sencillamente localizar una posición con los valores más similares a los recibidos pero nos lleva al reto de poder diferenciar dos posiciones con unos valores similares.

Se han encontrado varios métodos para decidir la localización a la que pertenece el patrón:

- **Producto escalar:**

Siendo en apariencia el método más sencillo consiste en convertir el vector de potencias recibido en un vector unitario y multiplicarlo por el vector unitario correspondiente a las localizaciones. Aquel producto cuyo resultado sea 1 o el más cercano posible sería el elegido.

- **Knn vecinos más cercanos**

En este caso también compararemos los vectores pero lo haremos de otra manera. Calcularemos la "distancia" entre nuestro vector y los que ya tenemos en la tabla y aquel con menor distancia será el elegido.

El concepto de distancia en este caso se determinará por la diferencia al cuadrado entre potencias:

$$D_i = \sqrt{(X_i - X)^2 + (Y_i - Y)^2 + (Z_i - Z)^2 + \cdots}$$

Y así con todos los valores del vector.

Al haber una cierta relación con la distancia y la potencia recibida de este modo se penaliza más que un valor se aleje más del que aparece en las medidas previas.

Este algoritmo es bastante más fiable que el anterior y sigue sin suponer una gran carga computacional para el dispositivo. Por contra tiene el inconveniente de que las tablas en las que se almacenarán las mediciones preliminares serán extensas.

Pese a ello será el método elegido por su simplicidad y precisión.

- **Métodos bayesianos**

El método conocido como Nibble es una mejora del anterior que construye una red bayesiana con todos los puntos.

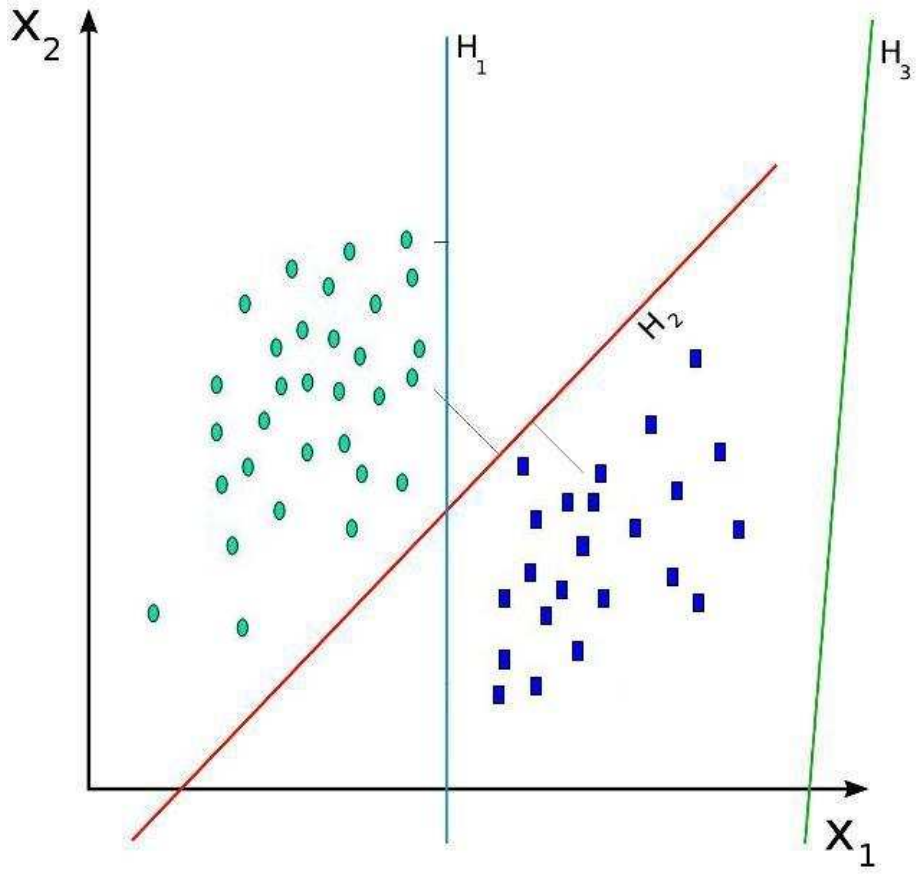
Este método se basa en las probabilidades de que el usuario se mueva a un determinado punto o permanezca en su posición. Con esos datos y las medidas de potencia tomadas llega a la conclusión de cuál es la posición más probable.

La calibración de dicho método puede llevar mucho tiempo y es más eficaz cuando se establece para un perfil de usuario concreto. Es decir, las probabilidades almacenadas no serían las mismas para dos usuarios distintos con unos patrones generales de movimiento distintos.

Este modelo también trata las transiciones imposibles entre localizaciones, lo cual puede ayudarnos a eliminar buena parte de la carga computacional al realizar los cálculos de la posición. Es similar a un Modelo Oculto de Markov.

- **Support Vector Machines**

En estos métodos se procesan los vectores de observación en un espacio con una mayor dimensión que la de las observaciones. De este modo se obtiene un hiperplano que separe las observaciones linealmente y permita la localización de la forma más fiable posible.



En resumen se trata de conseguir una frontera clara entre los posibles valores que pueden recibirse en una localización y en otra. Esa frontera es la encargada de decidir donde se encuentra el usuario.

Por desgracia aunque los resultados acostumbran a ser mucho mejores que con los métodos anteriores, requiere de una calibración muy laboriosa y son de compleja implementación.

2.3 Entornos adaptados para invidentes mediante localización

Aunque aún no hay demasiados, podemos encontrar algunos que nos muestran ideas muy interesantes acerca de cuantas formas puede facilitarse la vida del público objetivo.

2.3.1 Carro de la compra inteligente

Se trata de un carro de la compra con un dispositivo que indica al usuario el recorrido que ha de realizar, la localización de los productos en los estantes y le permite conocer los precios.

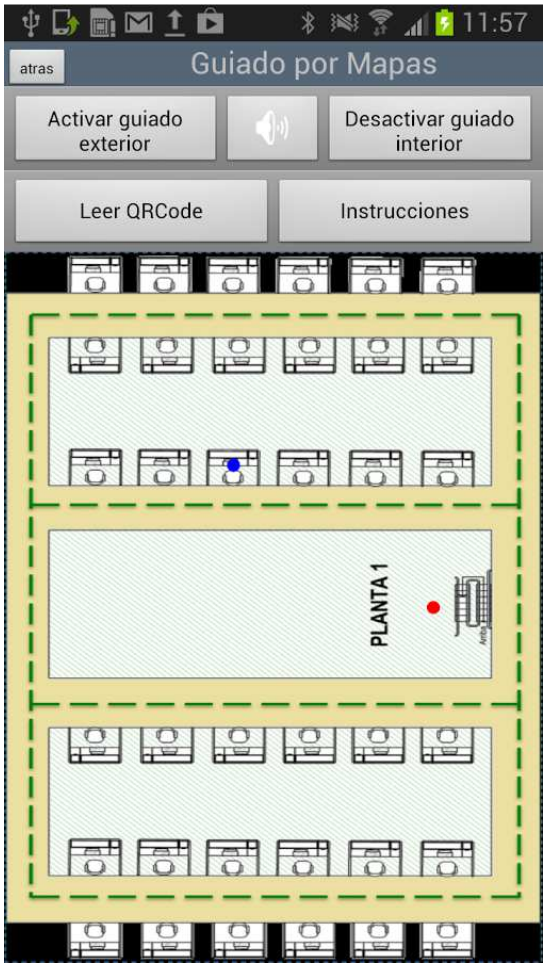
2.3.2 OnTheBus

Ya en Android nos encontramos con esta aplicación diseñada tanto para personas invidentes como con otras discapacidades que les permite desplazarse por la ciudad y utilizar el transporte público.

En sí consiste básicamente en una guía para llegar del punto A al punto B y ya funciona en ciudades como Barcelona, Madrid, Roma, Valencia y Zaragoza.

2.3.3 Ver con las manos

Este proyecto, que fue presentado por Cristina Rodríguez de la Universidad Rey Juan Carlos, ha presentado la aplicación "GuideURJC". Dicha aplicación permite la movilidad por todo el campus de la universidad a sus usuarios al combinar guiado tanto "Outdoor" como "Indoor" y les ofrece indicaciones dependiendo de la incapacidad que presente (voz para invidentes, gestos y texto para sordos, etc).



De paso sirve también de guía para todo aquel que quiera conocer la zona, así como puntos de interés.

Dicha aplicación combina varios sistemas de posicionamiento incluyendo uno basado en códigos QR colocados por los edificios. Además en caso de emergencia es capaz de indicar al usuario la salida de emergencia más cercana.

2.4 Conclusiones sobre el Estado del Arte

Tras haber contemplado las principales tecnologías y técnicas existentes para la localización en interiores y guiado de personas invidentes se han llegado a las siguientes conclusiones:

- Las tecnologías inalámbricas más sencillas de implementar para una aplicación de Android serían aquellas basadas en Bluetooth (concretamente BTLE) y los protocolos 802.11 dado que podemos encontrarlas ya presentes en una amplia variedad de dispositivos.

- Para la localización es más sencillo usar el sistema basado en reconocimiento de patrones RSSI dadas las dificultades que implican una localización en interiores.

- Será muy importante que la aplicación sea capaz de transmitir información y ser utilizada por un usuario invidente por razones obvias. Para ello habrá de implementar al menos algún tipo de comunicación oral con el usuario.

3. Propuesta de Entorno Inteligente Adaptado

3.1 Definición

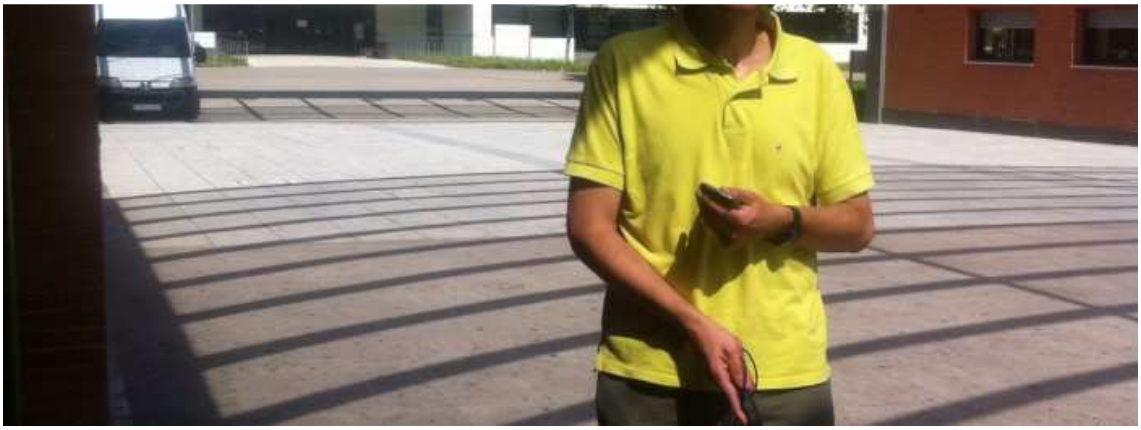
Nos referimos a un Entorno Inteligente como aquél en el que se han integrado suficientes dispositivos inteligentes que trabajan de manera continua para hacer la vida más cómoda a sus habitantes.

Estos entornos tratan de satisfacer la experiencia de las personas sustituyendo trabajo peligroso, físico o tareas repetitivas mediante agentes automatizados. Así mismo pueden reaccionar a los usuarios y a sus necesidades en tiempo real.

Cuando nos referimos a un Entorno Inteligente Adaptado lo hacemos porque queremos que esté adecuado concretamente a un tipo de usuario. En nuestro caso, como ya se ha dicho, a los usuarios invidentes o con graves problemas de visión.

3.2 Objetivo

El objetivo es concretamente conseguir adaptar un espacio tan común como una superficie comercial con productos de consumo diario como podría ser un supermercado a un público que se haya visto particularmente en desventaja en tales espacios como son los invidentes.



Para dicho entorno el usuario deberá ser capaz de haber creado una lista básica con los productos que desea y al activar la opción de guiado de la aplicación ésta deberá ser capaz de llevarle hasta cada producto siguiendo un recorrido que no le haga perder demasiado tiempo ni pasar varias veces por el mismo punto alargando el trayecto innecesariamente.

De este modo el entorno una vez adaptado mediante las balizas y la aplicación deberá poder ser accesible al usuario sin que éste dependa de la ayuda de otra persona.

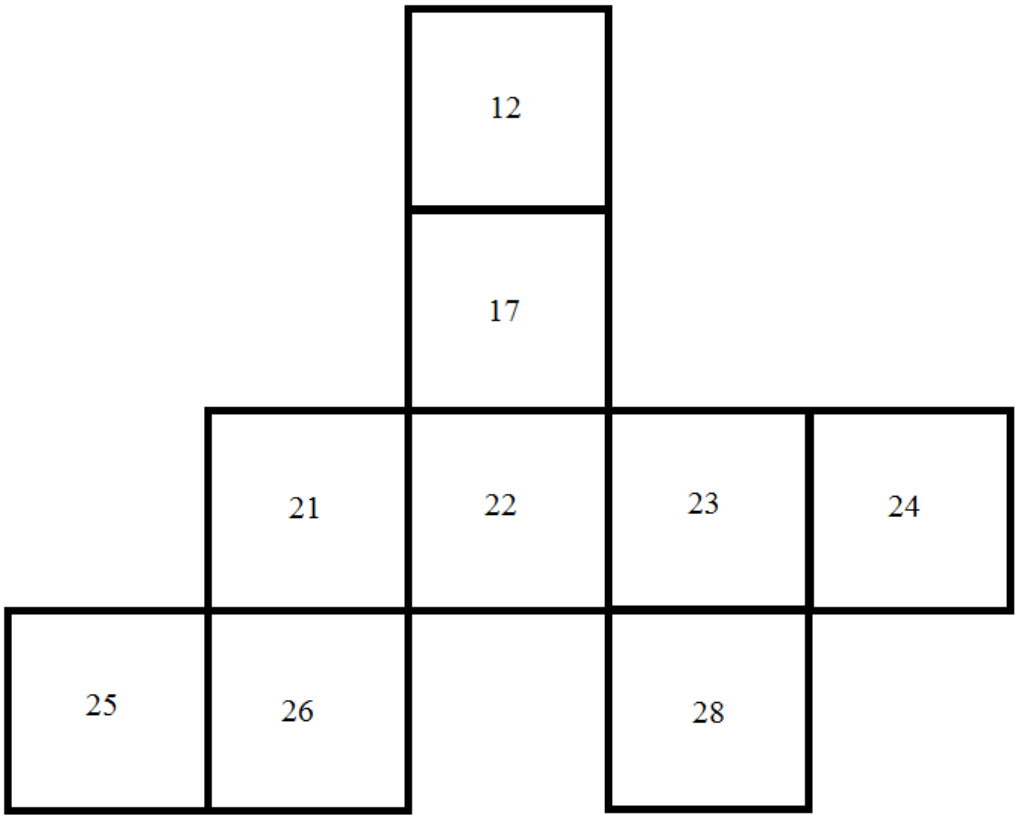
3.3 Entorno Inteligente Adaptado simulado

El entorno inteligente simulado fue un pequeño comercio con pasillos. En él se simulaba la existencia de 10 productos diferentes en distintas posiciones colocados de manera arbitraria.

Espacio físico: el espacio físico real en el que se realizaron las pruebas fue parte de una vivienda. De este modo los tabiques y otros elementos podían servir para comprobar los efectos que se esperaban sobre la señal por parte de un espacio cerrado.

División del espacio en sectores: La división se realizó en sectores cuadrados aunque no fueron todos del mismo tamaño. En total se dividió el espacio en 9 puntos de control que abarcaban aproximadamente 25 metros cuadrados de la vivienda.

De media dichos puntos de control abarcaban una superficie de 1.5 metros cuadrados, aunque algunos abarcaron una superficie mayor al quedar lejos de las balizas y no poder obtener por ello una gran precisión. Se procuró que los puntos de control estuviesen determinados por 1.5 metros aproximadamente dado que es la precisión que cabría esperar mediante los sistemas elegidos.



Velocidad de movimiento del usuario: Dado el público principal al que va destinada la aplicación no puede aproximarse el estándar habitual de movimiento. Una persona normal camina a una velocidad de entre 2 y 3 kilómetros por hora o entre 0.55 y 0.83 metros por segundo. Aún teniendo en cuenta la condición del usuario objetivo, por si acaso, tomaremos como velocidad 0.55 metros por segundo. Ello nos lleva a que el usuario no debería ser capaz de desplazarse más allá de 1 punto de control en 2 segundos, lo cual nos resultará útil realizando el algoritmo de localización.

3.4 Las Balizas

Para poder desarrollar el entorno fueron necesarias balizas BTLE proporcionadas amablemente por Embeblue. Dichas balizas actúan como dispositivos Bluetooth Low Energy con los que no es posible conectar pero que son detectables por el dispositivo móvil.

En este caso se contó con dos balizas que presentaban una distinta potencia de señal en condiciones similares. Dado que nuestro algoritmo no se basa en triangulación este hecho afortunadamente no supone ningún problema.

Además de las balizas, incluiremos en esta sección un router inalámbrico cuya señal fue aprovechada como tercera baliza.

De este modo se cubrió una superficie de unos 20 metros cuadrados con unos errores de localización relativamente bajos.

3.5 Dispositivo móvil

Como dispositivo móvil se utilizó un Smartphone con Android Samsung Galaxy S4. Aunque es un dispositivo considerado de gama alta el requisito indispensable para el uso de la señal BTLE no es la gama del dispositivo sino que este cuente con una versión de Android 4.3 (API Level 18) o superior. Por ello, aunque no sea de gama alta deberá tener soporte para dicha versión o al menos capacidad para que sea instalada.

Por desgracia en este dispositivo concreto la brújula se encontraba inutilizada y era un elemento vital para las indicaciones, por lo que se tuvo que recurrir a una solución en el propio software de la aplicación para detectar la dirección actual del usuario.

4. Desarrollo de la aplicación

La aplicación que será desarrollada concretamente para el dispositivo indicado, pero que es de esperar que sea compatible en otros, habrá de contar con los siguientes requisitos:

1. Capacidad para almacenar información: Deberá ser capaz de almacenar unas listas creadas por el usuario.

2. Búsqueda y detección de dispositivos: La aplicación debe ser capaz de localizar las balizas y detectar la potencia de señal recibida.

3. Identificación de dispositivos: Además debe de ser capaz de distinguir la potencia de cada dispositivo para añadirla al vector de potencias recibidas de modo que quede asociada con su respectivo emisor. Además los dispositivos ajenos a la aplicación no deberán influir en su funcionamiento.

4. Creación de rutas: Con la lista creada por el usuario deberá ordenar los productos por proximidad e idear una ruta a través de los puntos que componen el mapa hasta cada uno de los objetivos y volver a la salida.

5. Comunicación con el usuario: Debe de ser accesible tanto como para un usuario cualquiera como para uno invidente, por lo que deberá ser capaz de crear voz a partir de un texto dado.

6. Localización: La aplicación ha de poder deducir su posición de entre una serie de puntos creados en el mapa a partir de las potencias recibidas.

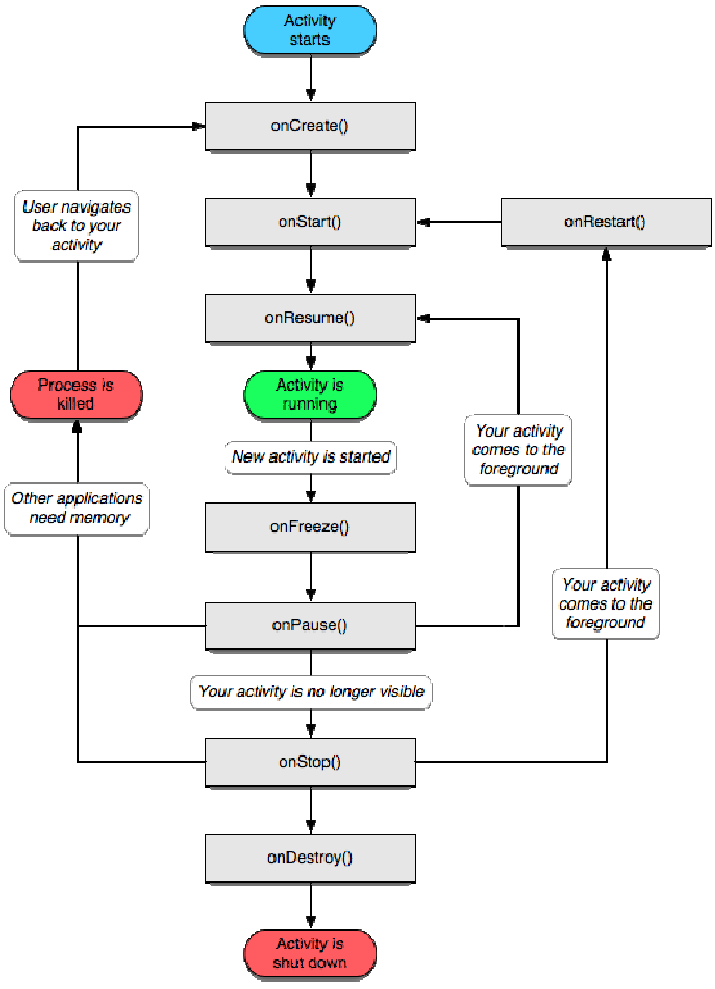
7. Capacidad de guiado: Basándose en la localización actual, la orientación y el siguiente objetivo, debe poder dar indicaciones al usuario para alcanzar el siguiente punto.

4.1 Diseño e implementación

La aplicación fue desarrollada poco a poco diseñando cada una de las Actividades que la componen de forma independiente y tratando después de relacionarlas entre sí.

Entre otros aspectos esto trajo consigo el tener que aprender a trabajar con los ciclos de vida de cada actividad en Android. Dichos ciclos hacen que no siempre al cambiar de actividad se cierre la anterior pudiendo volver a ella.

A causa de ello en varias aplicaciones se añadió una cantidad considerable de código en los métodos "OnResume()" y "OnPause()" para finalizar algunas funciones que no se reanudarían hasta volver a la propia actividad. En otros casos sencillamente se llamó al método "finish()" al entrar en el método "OnPause()" para terminar la actividad y que no se pudiese volver a ella.



4.1.1 Almacenamiento de la información

En un primer lugar se trató de que funcionase con una lista única que era guardada en las preferencias de la aplicación y después se mejoró para que admitiese más de una lista y el nombre de la lista a tratar se convirtió en un parámetro que habían de pasarse de unas Actividades a otras.

Lamentablemente no se podían crear las listas y acceder a ellas libremente desde cualquier otra Actividad. Además los métodos para pasar información entre dos Actividades hacen que ambas actividades tengan que ser ejecutadas una tras otra, lo cual no era lo que se precisaba. Se requería poder almacenar la lista y acceder a ella en cualquier momento.

Así pues, el almacenamiento de la información se realizó mediante las preferencias usando la clase SharedPreferences. En dichas preferencias pueden almacenarse pequeñas cantidades de información de tipos primitivos (Enteros, booleanos, texto, etc...).

Por desgracia era complicado trabajar con los datos de la lista de la compra en un formato tipo "String", por lo que se creó la clase "apoyo" en la que se almacenaron algunos métodos para convertir el String en un ArrayList y viceversa. De este modo podían añadirse o eliminarse objetos de la lista, así como ordenarlos mucho más fácilmente.

4.1.2 Búsqueda y detección de dispositivos.

Para realizar esta actividad se recurrió a un bucle que ejecuta cada 3 segundos. Al ejecutarse activa la búsqueda de dispositivos durante 2 segundos y cada vez que se detecta uno se recoge su señal, se compara su dirección con las almacenadas en la lista, se almacena la potencia en un vector y al final del periodo de escaneo se crea un vector de potencias con la media de las potencias recibidas durante dicho periodo de tiempo.

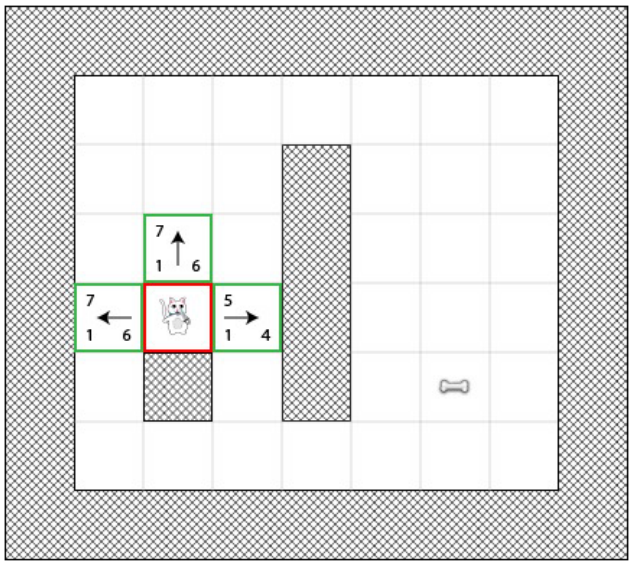
Se ha decidido que se recoja la media de la señal y no la señal instantánea porque se detectaron durante las mediciones unas variaciones muy importantes. Sin embargo al realizarse las mediciones almacenando la media en lugar del valor instantáneo se consiguieron suavizar considerablemente esos fallos de medición.

Para elegir el tiempo de escaneo fue determinante la velocidad del usuario. Si se desplazase más deprisa hubiese habido que reducir los periodos, pero conociendo a priori la velocidad a la que se supone que se moverá puede apurarse el tiempo de medición para conseguir una media más fiable.

4.1.3 Creación de rutas

Esta tarea resultó algo más complicada y se recurrió al algoritmo "A*" para localizar el camino más corto entre dos puntos sobre un mapa con obstáculos.

Dicho algoritmo combina muy bien sencillez y eficacia y dada su baja complejidad permitiría ser implementado relativamente rápido y no consumiría grandes recursos al ejecutarse.

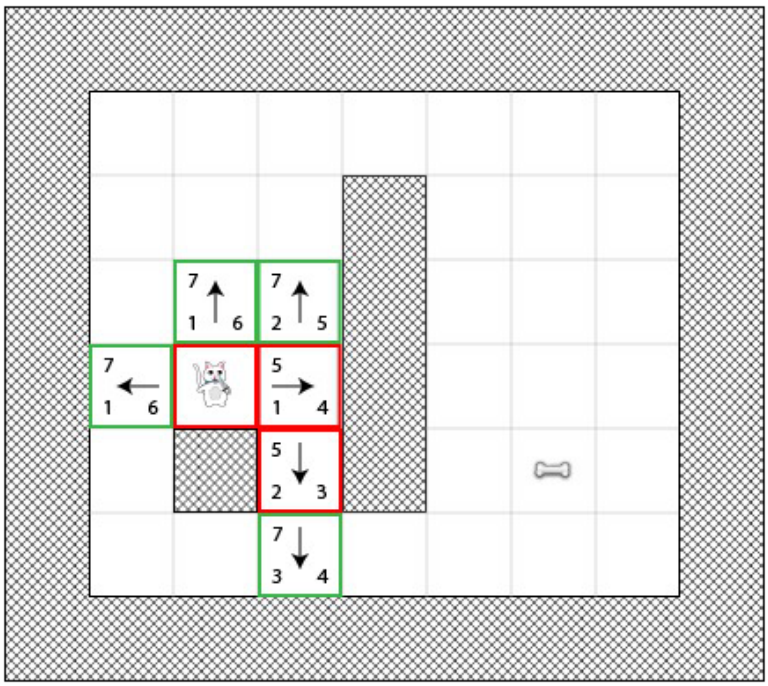


El primer paso para el desarrollo del algoritmo era la simplificación del área de búsqueda. Para ello se dividió el mapa en celdas cuadradas de un tamaño aproximado de 1.5 metros. El tamaño se debe tanto a la precisión estimada como a la combinación de tiempo de escaneo y velocidad supuesta del usuario.

Se creó con este fin la clase "Buscamino", en la que se encuentran almacenados varios métodos asociados con la localización de un camino entre dos puntos. También se utilizó la clase "Punto", que además de representar a cada posible localización permitía guardar los vecinos accesibles desde dicha localizacion.

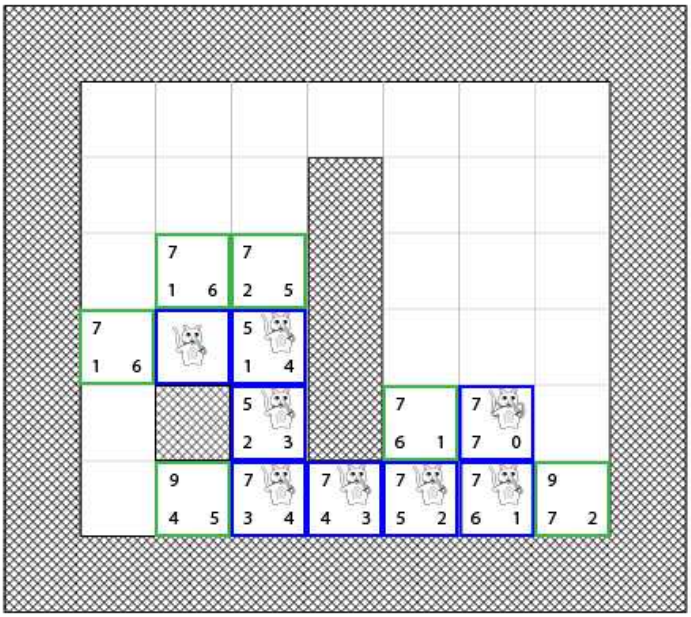
Con ello se procedió a implementar el algoritmo "A*":

Para que funcione se crean dos listas, una "abierta" y otra "cerrada" y se añade a la lista abierta nuestro punto inicial. A cada vecino posible se le asignan dos valores denominados G y H, siendo G el coste en desplazarse hasta ese vecino desde la casilla inicial y H una estimación de la distancia que aún separa a dicha casilla con la casilla final. En las imágenes de muestra puede apreciarse G en la casilla inferior izquierda de los cuadros y H en la derecha. La suma de ambos aparece en la parte superior.



Entonces pasamos la casilla inicial a la lista cerrada, sacándola de la abierta. Al hacerlo incluimos sus vecinos en la lista abierta. Después, de entre los vecinos, elegimos aquel cuya suma de G y H sea menor y lo trasladamos a la lista cerrada añadiendo sus vecinos a la abierta. Repetimos el proceso hasta que lleguemos a la casilla objetivo.

Es entonces el momento de recorrer la lista cerrada en orden inverso desde el objetivo eligiendo siempre una casilla con un coste menor (G) y que sea vecina de la casilla anterior. De este modo aunque hubiésemos añadido unas casillas pertenecientes a una bifurcación que acabó en un callejón sin salida evitaremos añadirlas a la lista definitiva de puntos a recorrer.



Una vez implementado el algoritmo que localiza el camino más corto entre dos puntos necesitamos crear una ruta que transcurra por todos los puntos especificados en la lista creada por el usuario.

Se estudiaron para ello varios algoritmos y se optó por la opción de menor coste computacional que acostumbraba a devolver un camino suficientemente corto visitando todos los puntos de interés sin repetir ninguno (como en el famoso problema del comerciante viajero).

Dicha solución consistía en moverse al punto más cercano, para lo cual desde la posición actual se calculaba la distancia más corta al resto de puntos por visitar. Una vez localizado el más cercano, se calculaba el siguiente más cercano a éste y así se iban eliminando de la lista hasta que tan sólo quedase la salida del recinto como obetivo.

Todo este algoritmo se creó basándose en un mapa compuesto por una tabla de unos y ceros, siendo los 1 camino que podía ser recorrido y los ceros los que no (muros, etc). En base a dicha tabla el algoritmo es capaz de deducir los vecinos a los que se puede acceder desde cada punto y asigna una ID para identificar a cada uno de manera inequívoca. La ventaja es que de este modo se adapta a cualquier mapa que se le quiera indicar y en el que se le asigne una posición a los distintos productos, pero la desventaja es que si dos puntos contiguos están separados, por ejemplo por un tabique, la aplicación no lo reconocerá y tratará de que el usuario atraviese dicho obstáculo si lo creyera necesario.

4.1.4 Comunicación con el usuario

Para resolver este punto se ha optado por utilizar una fuente suficientemente grande por si el usuario tan sólo tiene alguna dificultad en su visión.

Pero la cosa no ha quedado ahí. Se han diseñado todas las actividades de modo que los botones que la hacen funcionar ocupen toda la pantalla y si el usuario deja pulsado un botón este le leerá el texto que en él se halla.

Además en la actividad de guiado aparte de mostrar la indicación a seguir por pantalla también la lee una voz cada vez que la indicación a seguir cambia.

4.1.5 Localización

Como ya se ha mencionado antes el algoritmo utilizado para la localización se basa en el Knn vecino más cercano. De este modo al comparar el vector de potencias con los almacenados se consigue un porcentaje de acierto bastante elevado.

Se realizaron pruebas también con el método del producto vectorial pero los resultados obtenidos fueron mucho peores.

Una mejora añadida al método de localización y que también se basa en la supuesta velocidad del usuario es que la aplicación asume que se comienza en la entrada del establecimiento, lo que le da una posición inicial.

Con dicha posición la aplicación asume que le es imposible desplazarse a cualquier otro punto que no sea un vecino de la posición actual, lo cual limita la tabla de vectores a comparar tan sólo a los valores de sus vecinos y de la propia posición actual. De este modo se reduce significativamente la tasa de error por localizaciones que poseen unos vectores de potencias similares.

Las pruebas han verificado además que tras cada comprobación en caso de haberse desplazado más de lo estimado el usuario los valores de los vecinos más próximos a su posición real hacen que el algoritmo vaya desplazando la posición hasta donde éste se encuentra.

4.1.6 Capacidad de guiado

Para llevar a cabo este punto se requerían las rutas a seguir, la localización del usuario y la capacidad de comunicar las indicaciones.

La aplicación genera una lista con los puntos a seguir para desplazarse por el establecimiento y los compara con la posición actual del usuario. De este modo le guía hacia el siguiente punto en su lista.

Si el usuario sigue la lista, cada vez que haya un cambio de dirección o que se encuentre en la posición de alguno de los productos elegidos recibirá una indicación verbal avisándole de ello.

Si por el contrario se desplaza a un punto que ni es vecino del siguiente a recorrer ni es el punto hacia el que la aplicación le guiaba, ella misma recalculará todas las rutas a recorrer y le guiará por el nuevo camino a seguir.

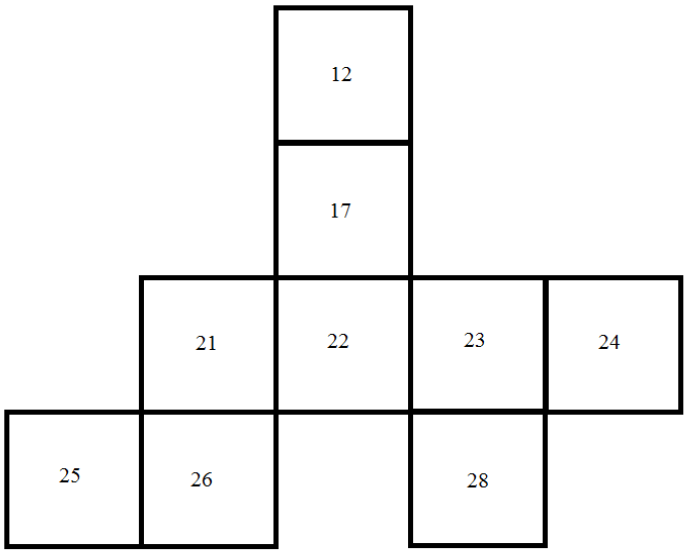
Cada vez que recorra un punto de interés dicho punto será eliminado de la lista, por lo que si se recalcula el recorrido la aplicación no hará volver al usuario a por un producto que ya ha recogido.

Para poder dar una indicación correcta al no contar con la brújula fue preciso crear un algoritmo que calculaba la orientación del usuario basándose en su último desplazamiento. Una vez obtenido dicha orientación, ésta era comparada con la que debía adoptar el usuario para seguir la trayectoria generada por la Aplicación y mediante comparación ésta le transmite indicaciones como "De media vuelta y avance." o "Gire a su izquierda y avance".

4.2 Preparación del Entorno Inteligente Simulado

Para poder utilizar la aplicación será necesario preparar un entorno en el que ésta pueda funcionar. Para ello eran necesarios varios pasos:

- Preparar la tabla que servirá de mapa
- Asignar posiciones a los productos, entrada y salida.
- Realizar varias medidas en cada punto para determinar su vector de potencias



4.2.1 Preparar la tabla que servirá de mapa

Para este caso concreto la tabla fue la siguiente:

0	0	1	0	0
0	0	1	0	0
0	1	1	1	1
1	1	0	1	0

Después, para poder añadir los vectores a sus respectivos puntos se calculó de antemano cual será su ID:

0	0	12	0	0
0	0	17	0	0
0	21	22	23	24
25	26	0	28	0

4.2.2 Asignar posiciones

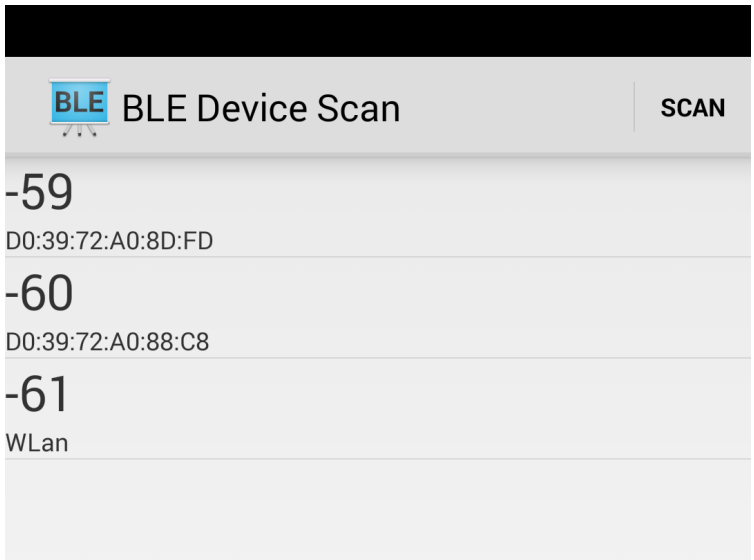
Se realiza en el código, en la clase "Posiciones" en la que también se define el mapa anterior. Para ello hay que tener en cuenta que el primer valor del par corresponde a la coordenada "x" siendo esta la posición vertical, que comienzan en "0" y que los valores para "x" comienzan en la parte superior aumentando según nos desplazamos hacia abajo.

```
public static final HashMap<String, int[]> lugar;
    static
    {
        lugar = new HashMap<String, int[]>();

        int[] actual = {2, 1}; //inicializado en la entrada
        lugar.put("actual", actual);

        int[] entrada = {2,1};
        lugar.put("Entrada",entrada);

        int[] salida = {2,1};
        lugar.put("Salida",salida);

        int[] pollo = {3,0};
        lugar.put("Pollo",pollo);
```

```
        int[] ternera = {3,1};
        lugar.put("Ternera",ternera);

        int[] zanahoria = {2,4};
        lugar.put("Zanahoria",zanahoria);

        int[] puerro = {2,3};
        lugar.put("Puerro",puerro);

        int[] leche = {3,1};
        lugar.put("Leche",leche);

        int[] yogurt = {3,1};
        lugar.put("Yogurt",yogurt);

        int[] manzana = {0,2};
        lugar.put("Manzana",manzana);

        int[] platano = {1,2};
        lugar.put("Platano",platano);

        int[] salmon = {2,2};
        lugar.put("Salmon",salmon);

        int[] sardinas = {2,2};
        lugar.put("Sardinas",sardinas);
    }
```

4.2.3 Realizar las medidas

Para poder hallar el vector de potencias correspondiente a cada punto se realizaron varias medidas en ellos. Fue necesario crear una segunda aplicación que escanease en busca de dispositivos, los identificase y ponderase los valores recibidos a lo largo de 10 segundos en cada medición.

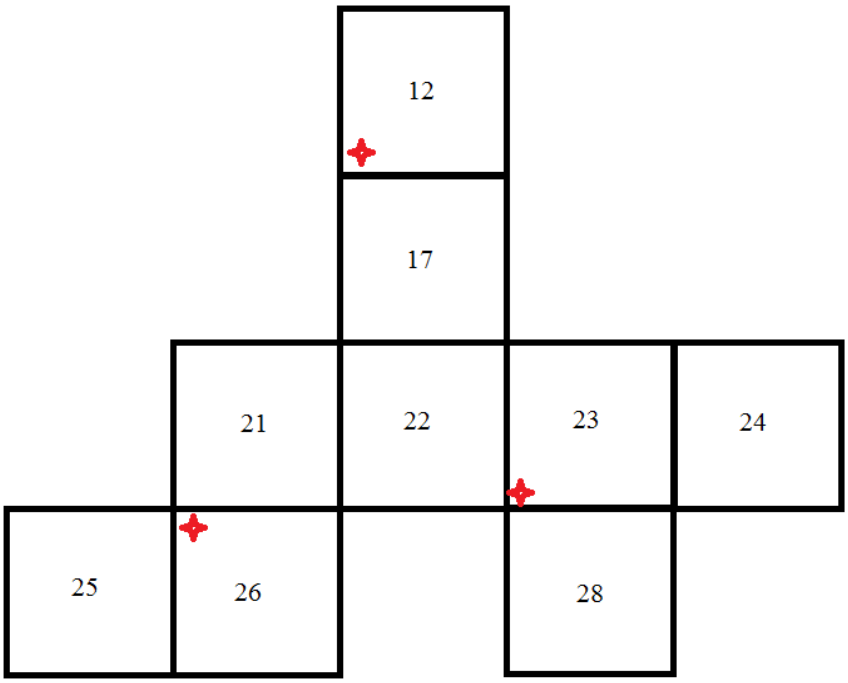


Para conseguir unos resultados más o menos fiables se repitieron las mediciones en más de 5 ocasiones sosteniendo el dispositivo siempre de la misma manera pero cambiando la orientación del usuario, dado que no siempre va a acceder al punto con la misma que con la que se realizaron las mediciones.

Los resultados fueron los siguientes vectores:

ID: 12	{-90, -58, -71}
ID: 17	{-83, -74, -63}
ID: 21	{-68, -84, -61}
ID: 22	{-80, -72, -63}
ID: 23	{-85, -83, -56}
ID: 24	{-86, -85, -57}
ID: 25	{-74, -93, -78}
ID: 26	{-61, -87, -77}
ID: 28	{-85, -86, -55}

Obtenidos por una distribución de las balizas de la siguiente forma:



5. Resultados obtenidos

5.1 Manejo de la aplicación

La aplicación no es intuitiva del todo para un usuario que no sepa cómo usarla. Necesitaría unas breves instrucciones. Después de ello un usuario puede desenvolverse sin demasiada dificultad a través de ella gracias a las indicaciones verbales obtenidas al dejar pulsados los botones.

5.2 Localización

Como era de esperar al contar con sólo 3 balizas la precisión no fue perfecta. Se dio el caso de un punto cuyas medidas fueron demasiado similares a las de dos de sus vecinos, por lo que aumentó considerablemente la tasa de error.

Para cada punto se creó la tabla de referencia de potencias realizando mediciones durante un minuto en distintas orientaciones y obteniendo la media. Después se procedió a comparar medidas realizadas en los puntos de interés con sus respectivos vectores en 5 ocasiones distintas.

En la siguiente tabla pueden apreciarse las "Distancias" calculadas. Para cada fila se encuentran los valores de cada punto medidos desde el indicado en la fila. En amarillo puede apreciarse el valor asignado como localización, en rojo el valor que debería haber sido en caso de error.

	A	B	C	D	E	F	G	H	I
A	20	413,24	437,96	187,24	304,04	419,04	444,2	427,44	1395,72
A	12,6	344,64	383,76	377,04	411,44	270,84	294,8	263,24	1419,12
A	12,2	323,44	348,56	360,24	365,84	246,44	266	240,84	1355,12
A	46,4	445,24	476,36	132,84	323,24	525,44	559,4	549,84	1360,52
A	118,8	85,64	177,96	740,84	808,04	193,04	259,4	243,44	868,52
B	442,8	126,44	109,56	1273,64	902,84	29,84	48,2	62,24	712,52
B	158,2	54,24	138,16	817,04	842,64	165,24	230,8	221,64	771,12
B	165	68,24	51,36	500,24	438,24	204,84	239,6	277,24	511,92
B	358,4	43,64	70,76	823,24	834,44	335,04	403,4	451,44	289,32
B	668,6	98,24	168,56	1462,64	1413,84	411,64	502	544,04	275,92
C	594,4	96,44	78,76	1100,04	970,44	388,64	446,6	521,04	124,52
C	361	49,04	3,76	847,04	623,44	159,24	187,2	241,64	328,32
C	277,2	24,84	7,56	861,64	648,84	76,24	106,6	138,64	475,72
C	83,6	88,04	94,76	576,44	466,44	78,64	105,4	109,04	817,72
C	515,6	106,44	32,36	1102,84	735,24	142,64	156,6	217,04	344,12
D	280,4	979,24	987,56	38,84	344,84	1049,84	1068,6	1056,24	2024,12
D	391,8	968,64	863,76	37,84	91,44	963,64	941,2	972,04	1751,92
D	356,4	809,64	666,36	159,24	5,24	684,24	641	676,64	1566,12
D	462,2	1039,04	1020,56	37,84	401,84	1283,64	1312,4	1340,04	1745,52
D	484,2	1076,24	1119,36	113,04	626,24	1405,64	1462	1474,04	1844,72
E	478	742,44	547,96	402,44	34,04	543,44	481	535,84	1348,52
E	342,8	601,64	443,96	312,84	26,04	453,04	407,4	453,44	1236,52
E	708,6	1527,04	1362,96	227,44	157,04	1314,04	1247,6	1266,44	2546,32
E	308,6	643,84	511,76	152,24	19,44	597,24	567,6	613,64	1267,92
E	403,4	765,04	599,36	226,64	2,24	640,04	591,6	640,44	1412,72
F	355	126,24	61,36	968,24	588,24	30,84	31,6	63,24	643,92
F	177,2	96,84	122,36	873,64	693,24	29,84	59,4	50,24	890,12
F	470	148,04	146,36	1364,04	997,24	39,44	61,8	67,84	782,12
F	282,4	275,64	237,16	963,24	589,64	23,84	13,8	4,24	1168,52
F	299	247,84	199,76	969,84	579,44	14,84	4,4	3,24	1069,52
G	160,2	146,64	99,36	631,44	374,24	46,44	47,6	62,84	838,32
G	560	400,44	353,96	1458,44	932,04	69,44	53	41,84	1276,52
G	576,2	389,04	276,56	1283,84	689,84	81,64	46,4	68,04	1059,52
G	186,6	167,44	124,56	729,84	437,84	24,04	22,8	30,44	916,72
G	330,8	129,64	130,76	1134,84	822,44	10,64	30,2	29,04	852,92
H	255,6	233,64	176,76	848,04	480,44	21,84	9,8	14,24	1026,12
H	345,4	283,04	265,36	1142,64	760,24	26,04	23,6	6,44	1200,72
H	78,2	210,24	250,16	673,04	578,64	101,24	126,8	97,64	1211,12
H	699,2	509,24	382,76	1456,04	786,44	133,44	87,4	105,84	1222,92
H	391,2	184,44	189,96	1267,24	920,04	22,24	39,4	30,64	965,32
I	1526,8	711,24	595,96	1770,44	1562,04	1224,24	1277,8	1436,64	58,92
I	903,4	293,04	163,36	1474,64	974,24	364,04	367,6	464,44	236,72
I	1275	449,44	474,96	1847,44	1845,04	1024,44	1130	1236,84	65,52
I	1654,8	616,84	584,76	2364,04	2120,44	1109,04	1193	1321,44	63,72
I	1315,4	585,84	540,56	1575,44	1557,84	1180,04	1262	1398,44	61,52

De este modo podemos ver una precisión del 66%, pero si aplicamos el hecho de que tan sólo pueda elegirse la posición en la que se encuentra o un vecino sube hasta el 71%. Además el punto H fue problemático porque junto con sus dos vecinos más cercanos no tenían buena recepción de la señal de dos de las balizas al haber varios tabiques en medio. Si no lo tenemos en cuenta la precisión pasa a ser del 72% o del 75% aplicando la restricción de los vecinos.

Teniendo en cuenta el escaso número de balizas utilizadas y el gran número de obstáculos lo consideraremos un resultado aceptable. De hecho en un primer momento se trabajó con sólo dos balizas y la precisión fue bastante pobre. Cuando se procedió a añadir el router como tercera baliza se mejoró mucho.

5.3 Pruebas físicas

Con la colaboración de un sujeto de prueba se procedió a la creación y seguimiento de varias listas de la compra en dicho entorno.

El sujeto se desenvolvió bien hasta que se aproximó a la zona conflictiva del punto H. Allí las indicaciones comenzaron a cambiar continuamente al no decidirse la aplicación por la posición al ser similares ambos vecinos.

No obstante fue capaz de recorrer el resto de puntos sin mayor problema siguiendo las indicicaciones dadas por la aplicación.

6. Conclusiones y mejoras

6.1 Conclusiones obtenidas

Tras comprobar el funcionamiento de este proyecto tengo plena confianza en que los Entornos Inteligentes pueden ser una herramienta tremendamente valiosa para todos los ciudadanos, pero si además se diseñan para permitir la adaptación de personas con discapacidad para ellos pueden ser todo un regalo.

Apenas se ven hoy en día Entornos Inteligentes, y no digamos especialmente adaptados, pero poco a poco van llegando y no cabe duda de que en un futuro próximo los encontraremos en todas partes. Pronto puede que se conviertan en una nueva necesidad como lo es para la gente de hoy en día el estar siempre conectado o al menos con acceso a Internet.

La aplicación aquí desarrollada demuestra además que con unos pocos medios y algo de tiempo puede generarse un entorno adaptado a cualquier situación. En nuestro caso nos ocupaban los invidentes, pero el resto de usuarios también pueden beneficiarse de ellos así como las empresas ofreciendo pequeñas ofertas dependiendo de en que sección se encuentre el cliente, etc.

Podemos ver además como cada vez son más las grandes compañías como Google y Apple las que apuestan por el posicionamiento en interiores. Hace poco se empezaba a oír hablar de las iBeacon, las balizas de una de las compañías mencionadas y en internet pueden encontrarse ya varias ofertas para desarrolladores que quieran conseguirlas.

6.2 Mejoras a implementar

No son pocas las mejoras que se podrían implementar en esta aplicación, pero procederé a enumerar algunas de ellas.

- En primer lugar el sistema de almacenamiento de la información. Lo ideal sería trabajar con bases de datos como MySQL.

- Permitir conectividad a la aplicación para que pueda actualizar sus bases de datos si es que se cambian los productos de posición o si se llega a una superficie nueva con un mapa que aún no se haya descargado.

- Añadir información sobre la posición en los estantes del producto y que la aplicación la describa al usuario.

- Implementar la funcionalidad de leer códigos de barras y QR, así como colocarlos en los estantes junto a los productos. Esto permitiría al usuario identificar de una manera más rápida los productos al leerle la aplicación lo que se encuentra ante él y permitiéndole distinguir los productos una vez localizado el código de barras en caso de ser muy similares. Además esta función permitiría la calibración de la posición en el acto dado que cierto código QR sólo podría accederse desde una determinada posición.

- Implementar la funcionalidad de la brújula. Esto no sólo permitiría dar las indicaciones de una manera más fiable sino que nos otorga la oportunidad de no crear un solo mapa de valores RSSI. Tengamos en cuenta que el propio cuerpo del usuario atenuará la señal de algunas balizas, y conociendo su orientación pueden crearse al menos 4 mapas distintos para cada punto cardinal que será consultado en función de hacia dónde se encuentre mirando el usuario en ese instante.

- Sistema de emergencia. Si se ha habilitado la conectividad de la aplicación y se detecta una emergencia en el local en el que se encuentre el usuario, la aplicación deberá ser capaz de avisarle e indicarle la salida de emergencia más próxima. Así mismo podría implementar un sistema

de auxilio que transmita la posición del usuario a los responsables de seguridad del lugar.

- Localización en exteriores. Aunque ya existen aplicaciones para ello, podría prepararle la ruta hasta el supermercado más cercano que posea al completo los productos de la lista que el usuario seleccione y una vez dentro iniciar la compra.

- Mejorar la distribución sobre el mapa de las balizas para cubrir mejor la superficie. De hecho se han encontrado maneras de mejorar la precisión contando con balizas "virtuales" al relacionar mediante correlación la potencia de balizas en varios puntos, pudiendo prescindirse después de algunas de ellas. De este modo puede cubrirse una mayor área con una menor cantidad.

- El algoritmo para crear los recorridos puede ser mejorado también dividiendo el área en pequeñas zonas dentro de las cuales se realizaría un recorrido como el que se ha realizado hasta ahora. La diferencia estaría en que primero recorrería todos los productos de esa zona y ya no volvería a ella más tarde, lo cual puede resultar mucho más cómodo y agradable. Es decir, en lugar de acceder en dos ocasiones a un pasillo por sus extremos para recoger un producto en cada punta, recorrer todo el pasillo de una vez para no volver a él. Esto requerirá un estudio de cada zona de forma independiente.

- El algoritmo que genera el mapa de puntos que es posible recorrer y sus vecinos no tiene en cuenta tabiques o pequeñas estanterías. Está creado para funcionar con pasillos separados por anchas estanterías principalmente. Aunque el algoritmo actual permite adaptarse a cualquier espacio introducido con esas características sería recomendable que en versiones siguientes se estudiase más detenidamente el terreno a tratar.

- Se puede reducir el número de actividades creadas para añadir los productos si se crea una actividad base que se amolde a cada categoría de productos cargando los que requiriese en cada momento.

7. Presupuesto

Material

Nº de orden	Concepto/Referencia	Cantidad	Precio Unitario	Total
1	Smartphone con Android 4.3 o superior (Galaxy S4)	1	400€	400€
2	Balizas Bluetooth Low Energy	2	59€	118€
3	Router inalámbrico	1	15€	15€
4	Ordenador personal	1	500€	500€
	Subtotal			1.033€

Honorarios por mano de obra

Nº de orden	Concepto/Referencia	Cantidad	Precio Unitario	Total
5	H Ingeniero de Telecomunicación (Programación)	75	100€	7.500€
6	H Ingeniero de Telecomunicación (Mediciones, mapeado pruebas y simulaciones)	5	100€	500€
	Subtotal			8.000€

Total ejecución material y mano de obra ..... *9.033€*

Beneficio industrial (20%)..... *1.806,60€*

I.V.A. (21%) ..... *1.896,93€*

El **total** del presupuesto asciende a la cantidad de **doce mil setecientos treinta y seis Euros con cincuenta y tres céntimos (12.736,53€)**

Bibliografía consultada

Documentos

Javier García de Jalón, José Ignacio Rodríguez, Iñigo Mingo, Aitor Imaz, Alfonso Brazález, Alberto Larzabal, Jesús Calleja, Jon García - "Aprenda Java como si estuviera en primero"

Jorge Sanchez (2004) - "Java2"

Luis Díaz, Ambrona Tabernilla - TFC "Sistema de localización en interiores"

Adolfo González Blázquez, Pablo Pedro Mulas Gómez, Rafael Rivera Retamar - "Localización de dispositívos móviles en interiores usando redes Wireless"

Miguel Ruiz Cuesta - PFC "Posicionamiento en interiores basado en dispositivos móviles"

Javier A. Cuenca Retana (2010) - "Sistema de posicionamiento en interiores mediante balizas Bluetooth"

Jaime Alberto Guzmán Luna, Rafael Esteban Arango Sánchez, Leidy Diana Jiménez Pinzón (2012) - "Búsqueda de la ruta óptima mediante los algoritmos: genético y dijkstra utilizando mapas de visibilidad"

Eduardo Navarro, Benjamin Peuker, Michael Quan - "Wi-Fi Localization using RSSI Fingerprinting"

A. K. M. Mahtab Hossain, Hien Nguyen Van, Wee-Seng Soh - "Fingerprint-based Estimation with Virtual Acces Points"

F. Alsehly, R.Mohd Sabri, Z. Sevak, T. Arslan - "Improving Indoor positioning accuracy through a Wi-Fi Handover Algorithm"

Jungmin So, Joo-Yub Lee, Cheal-Hwan Yoon, Hyunjae Park - "Analysis of location estimation algorithms for wifi fingerprint-based indoor localization"

Jungmin So, Joo-Yub Lee, Cheal-Hwan Yoon, Hyunjae Park - "An Improved location estimation method for Wifi fingerprint-based Indoor Location"

Nissanka Bodhi Priyantha (2005) - "The Cricket Indoor Location System"

Jeff Hightower - "Mobile Location Technologies"

Fernando Seco Granja - Apuntes 2007/2008 Máster Oficial en Sistemas Electrónicos Avanzados. Sistemas Inteligentes. "Tema 3.3: Sistemas de localización en interiores basados en radiofrecuencia"

Ernst & Young - "Informe sobre la ceguera en España"

Cristina Rodríguez Sánchez - "Ver con las Manos: Plataforma MueveteURJC de Apoyo a Personas con Discapacidad para la accesibilidad y movilidad en entornos universitarios"

Páginas de internet

Android Developers
http://developer.android.com

stackoverflow
http://stackoverflow.com/

Wikipedia
http://www.wikipedia.org/

Universidad de Vigo: Método de localización en interiores basado en la fusión de tecnologías Bluetooth y Wlan
http://tv.uvigo.es/es/video/26858.html

Movilidad autónoma de invidentes en supermercados
http://prezi.com/ob_smqinh_b9/movilidad-autonoma-de-invidentes-en-supermercados/

Reivindican el derecho de los ciegos a hacer la compra
http://www.europapress.es/sociedad/noticia-reivindican-derecho-ciegos-hacer-compra-20131016101638.html

Sensores y una audioguía: un nuevo dispositivo para la orientación de invidentes
http://es.rbth.com/cultura/technologias/2013/10/21/sensores_y_una_audioguia_un_nuevo_dispositivo_para_la_or_33497.html

Embeblue
http://embeblue.com/

Sweet Beacon
http://www.sweetbeacon.com/

beMee
http://www.bemee.es/

Geolocalización en interiores, un futuro muy prometedor
http://salvadorvilalta.com/2012/08/26/geolocalizacion-en-interiores-un-futuro-muy-prometedor/

Xataka: La localización en interiores llega con la versión 6.0 de Google Maps
http://www.xatakamovil.com/aplicaciones/la-localizacion-en-interiores-llega-a-android-con-la-version-60-de-google-maps

Geolocalización de interiores: Se acercan grandes oportunidades para los negocios locales
http://juanchosierrar.blogspot.com.es/2012/12/geolocalizacion-de-interiores-se.html

Geolocalización en interiores
http://www.nosolosig.com/articulos/44-geolocalizacion-en-interiores

Orientar a los pasajeros mediante una aplicación de geolocalización
http://es.kioskea.net/faq/8874-aplicacion-de-geolocalizacion-en-interiores

Nokia introducirá posicionamiento por Bluetooth 4.0 en su servicio de mapas para edificios
http://es.engadget.com/2011/11/30/nokia-introducira-posicionamiento-por-bluetooth-4-0-en-su-servic/

El smartphone: un ojo para los invidentes
http://mobileworldcapital.com/es/articulo/114

Una aplicación que hace de lazarillo para los invidentes
http://www.20minutos.es/noticia/1760899/0/aplicacion-movil/lazarillo/ciegos/

"Beacons", su potencial y ventajas frente a la tradicional geolocalización por GPS
http://www.20minutos.es/noticia/2052004/0/beacons/mejor-que-gps/geolocalizacion/

Anexo: Administración energética en entornos urbóticos

A.1 ¿Qué es una "Smart City"?

Hoy en día el término "Smart City" sigue siendo más un concepto que una definición de cómo ha de ser una ciudad para ser considerada Inteligente. Cuando nos referimos a ello procuramos hacerlo como a un entorno urbano con un desarrollo basado principalmente en la sostenibilidad y que es capaz de responder de una manera adecuada a las necesidades básicas de instituciones, empresas y por supuesto las de sus propios habitantes.

Es por ello que lo que hace a una ciudad "Inteligente" no es el hecho de contar con las últimas tecnologías en ciertos servicios. Lo que la hace entrar en esta categoría es el ser capaz de haber utilizado esas tecnologías, u otras no tan punteras, para mejorar el comfort de sus ciudadanos brindando servicios de calidad mientras se respeta el aspecto ambiental y un uso prudente de recursos naturales no renovables.

En la práctica, a nivel popular, lo que se conoce como una "Ciudad Inteligente" es una ciudad comprometida con su entorno, con elementos arquitectónicos de vanguardia y con infraestructuras dotadas de soluciones tecnológicas avanzadas. Algunas de estas soluciones podrían ser, como en el ejemplo de Pamplona, una eficaz administración electrónica.

El concepto de "Smart City" es articulado en torno a las siguientes ideas:

- Cuestiones ambientales y restricciones energéticas.
- Comunicación fluida entre sus actores: colectivos, ciudadanos, empresas, instituciones...
- Uso compartido de bienes y servicios y la participación de los ciudadanos renunciando en ocasiones a la propiedad y uso individual.

- Integración de las nuevas tecnologías de la comunicación y de la información, robótica, sistemas inteligentes de transporte potenciando el funcionamiento en red, favorecimiento de las energías renovables y una concienciación por parte de los ciudadanos.

En la actualidad además no encontramos dos "Smart Cities" iguales. Esto se debe a que hay varios puntos en los que pueden centrarse.

Economía

Las que se centran en este apartado favorecen la modernización de empresas. En especial aquellas más influyentes en el desarrollo de la ciudad.

Movilidad

El apartado de movilidad se centra en servicios que hagan descender la densidad de tráfico urbano, tiempos de conducción y en la optimización de los servicios de transporte público para que estén mejor interconectados creando una red de transporte altamente eficiente.

Medioambiente

Con el punto de vista en la sostenibilidad y el poder llevar a cabo todo el desarrollo posible sin dañar o incluso favoreciendo a su entorno. Con servicios que maximizan la eficiencia del consumo de agua y de la gestión de residuos por ejemplo.

Calidad de vida

Favoreciendo un incremento en la seguridad de los ciudadanos y la coordinación de servicios sanitarios y de asistencia.

Administración energética

En ocasiones incluido junto con medioambiente, centrado en la reducción del consumo energético (electricidad, gas...) y de dar rápida solución a los cortes de suministro.

E-Gobierno

Servicios orientados a la modernización de la administración pública para asignar de una manera más eficiente los recursos de la ciudad así como para favorecer la transparencia y una administración electrónica.

Todo esto nos lleva a que la categoría "Smart" para una ciudad no es algo duradero. No es tan sencillo como conseguir unos objetivos sino que implica un compromiso con un proceso de mejora constante.

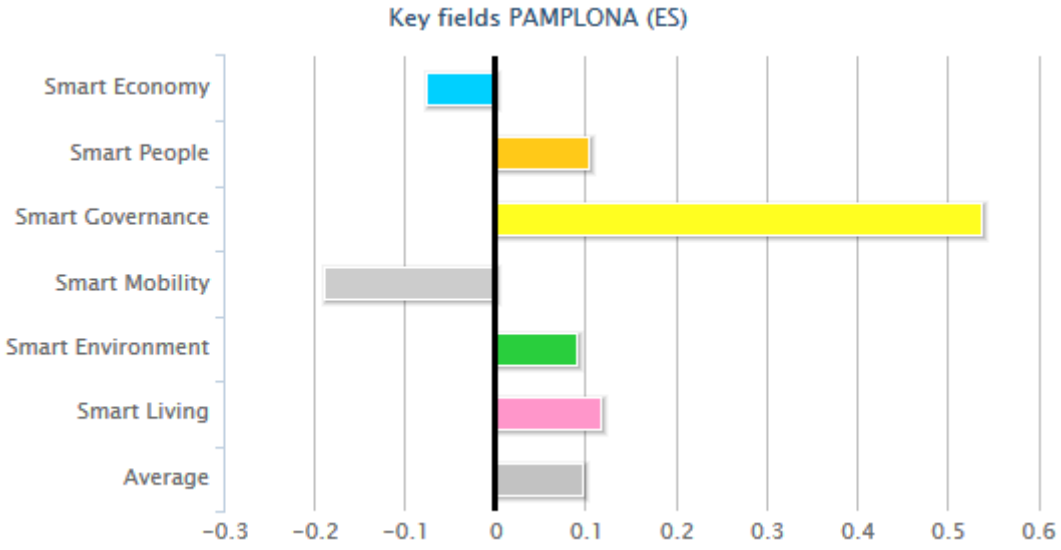
A.2 Estado actual

Pamplona

Actualmente Pamplona tiene una estrategia para ser considerada Smart City centrada en la calidad de vida, el servicio al ciudadano y la eficiencia en los procesos.

El seguir dicha estrategia le ha llevado a colocarse en la posición nº 38 del ranking de ciudades inteligentes europeas del 2014, lo cual implica un esfuerzo interesante ya que cuando entró en el 2007 se encontraba en el puesto 41 y como ya se

ha mencionado no es algo que implique sencillamente el lograr unos objetivos sino un compromiso de mejora.

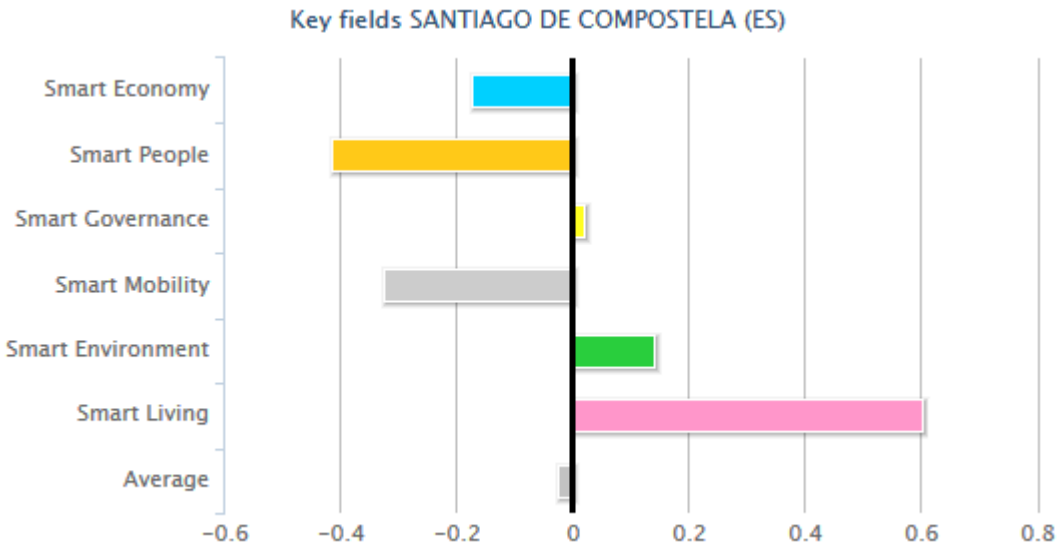


Como puede verse destaca especialmente por su gobierno, y no es de extrañar pues como hemos visto recientemente se han dado en Pamplona grandes avances en lo referente a administración electrónica. Avances como la factura electrónica, la posibilidad de llevar a cabo diversas gestiones a través de internet, etc. son detalles que facilitan la vida a sus ciudadanos y la hacen más eficiente.

Por contra como podemos apreciar peca de una tasa de movilidad algo baja, pero es la primera ciudad española en aparecer en el ranking.

Santiago de Compostela

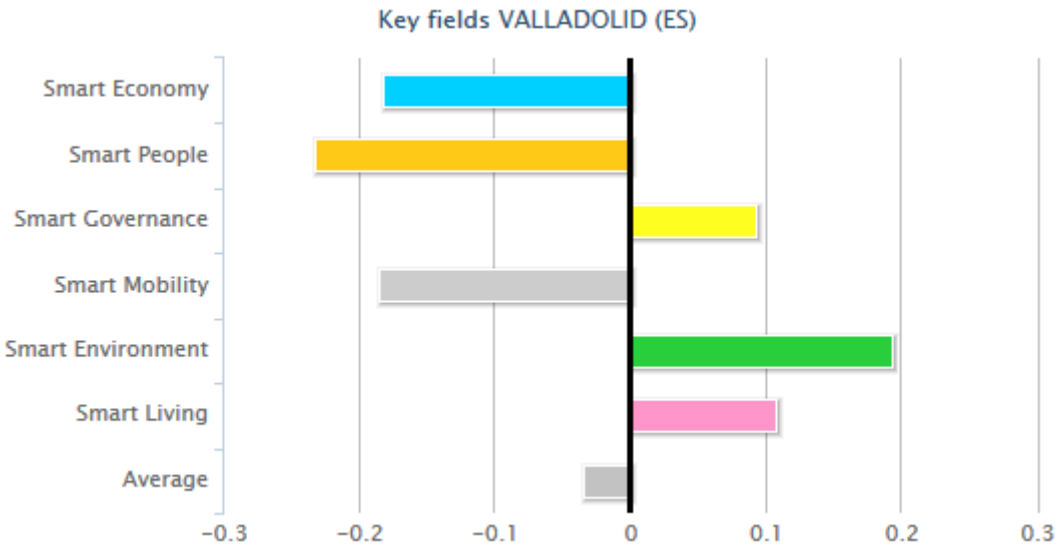
A nivel nacional, la segunda ciudad en aparecer en el ranking es Santiago de Compostela en la posición número 43.



Como podemos apreciar, esta ciudad no se ha centrado tanto en el gobierno sino en la calidad de vida y en el medioambiente.

Valladolid

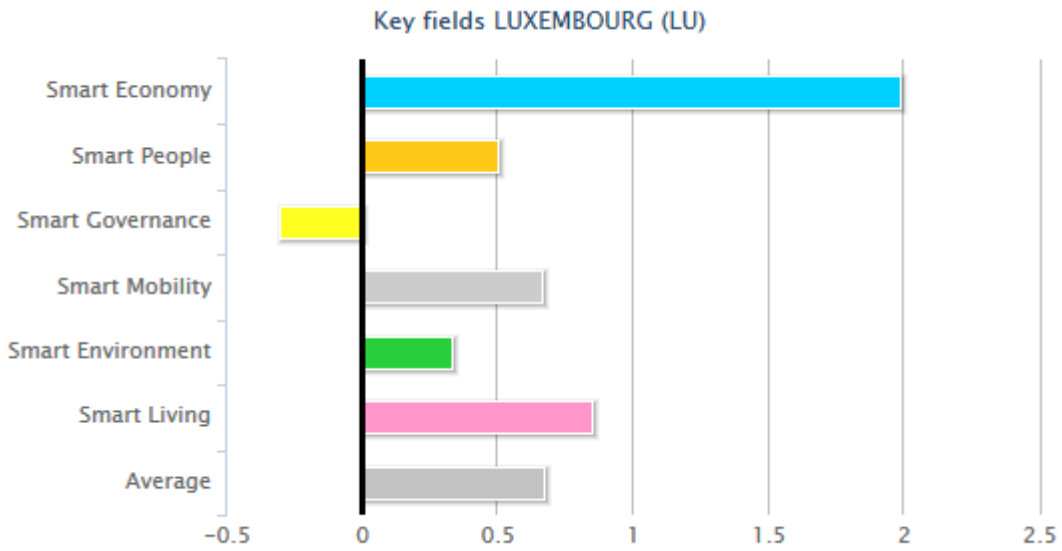
La tercera ciudad Española según dicho ranking sería Valladolid con el puesto 44, inmediata seguidora de Santiago.



En este caso podemos contemplar que sobresale especialmente en su relación con el medioambiente. Esto se debe a la implantación de una red inteligente de contadores, a su apoyo a la implantación del coche eléctrico y a su apuesta por la eficiencia energética en edificios singulares, la organización del tráfico para una mayor eficiencia y a su gestión de aguas residuales.

Luxemburgo

Luxemburgo ha sido la ciudad europea que ha recibido el puesto número 1 en lo referente a Smart Cities. Como podemos apreciar varias de sus puntuaciones son bastante buenas con un punto especial en el apartado económico.



A.3 Administración energética en entornos urbóticos

Si bien ya podíamos ver similitudes entre los 6 puntos centrales de las ciudades inteligentes y los de la domótica (confort, seguridad, ahorro energético y comunicaciones) cuando nos referimos a un entorno urbótico lo hacemos expandiendo el concepto de la domótica a toda una ciudad. Sin ir más lejos una de las definiciones encontradas es "Integración de la tecnología en el diseño inteligente de una ciudad".

A.3.1 ¿Qué es la administración energética?

Pese a que a priori parece una pregunta con una respuesta sencilla en el momento en el que nos paramos a pensar en un entorno urbótico lleno de sensores y actuadores capaz de controlarlo todo la respuesta se vuelve algo más difusa.

La administración energética no se reduce al control de las corrientes eléctricas en una ciudad cuando nos encontramos en un entorno urbótico, sino que pasa a ser algo más complejo y abstracto.

Si mediante los sensores detectamos que hay una gran cantidad de tráfico en unas determinadas calles, podremos modificar los ciclos de tiempo de los semáforos para hacerlas más fluidas. También podemos notificarlo a los usuarios de una aplicación, difundida previamente entre los conductores, para prevenirles de las zonas saturadas y sugerirles un camino alternativo por calles menos transitadas. Esto por ejemplo haría reducir el consumo de combustible por parte de los vehículos en nuestro entorno.

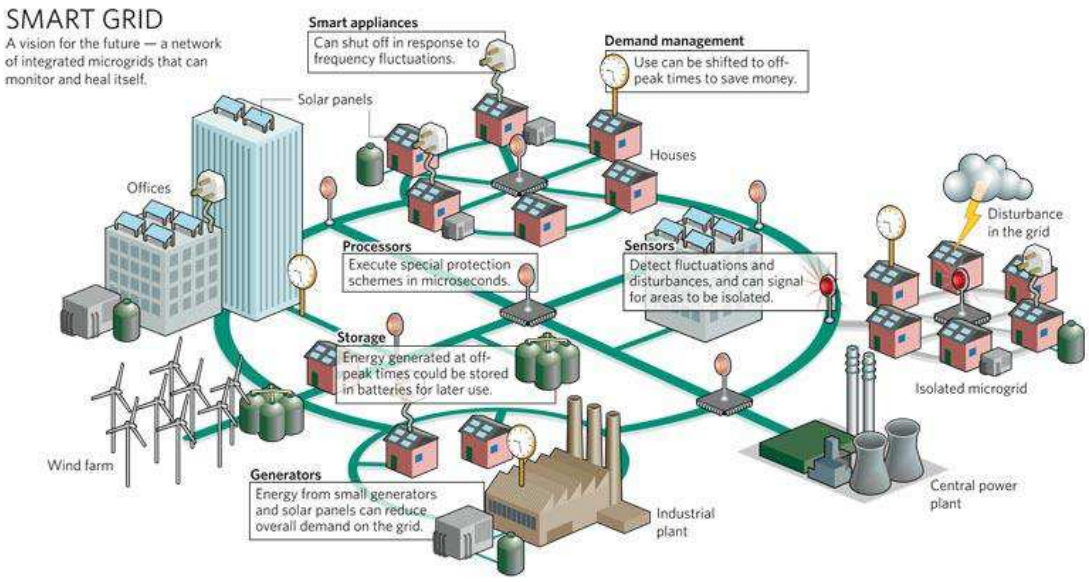
Si instalamos sensores de luminosidad en las calles y programamos bien las luminarias conseguiremos que éstas no estén encendidas antes de que sea necesario o que permanezcan apagadas minutos más tarde de pasar el umbral en el que deberían activarse. Si además les añadimos detectores de presencia como ya se ha hecho en varias ciudades podemos conseguir que las luminarias se coloquen en un consumo mínimo iluminando lo indispensable para aumentar la luz en caso de detectar una presencia y volviendo a su estado anterior cuando deje de notarla.

Estos dos ejemplos sirven para mostrar todo lo que puede llegar a abarcarse con un concepto tan simple cuando se posee un entorno tan complejo.

A.3.2 SmartGrid

Si nos decantamos concretamente por la red eléctrica podemos encontrar como solución a la administración energética las conocidas como "Smart Grids". Son una red de distribución de energía eléctrica inteligente capaz de reaccionar ante cambios en su entorno.

Una definición bastante acertada sería la de "Red que integra de manera inteligente las acciones de los usuarios que se encuentran conectados a ella (sean generadores, consumidores o ambos a la vez) con el fin de conseguir un suministro eléctrico eficiente, seguro y sostenible.



Por ello son unas redes que utilizarán equipos y servicios innovadores junto con nuevas tecnologías de la información, control, monitorización y auto-diagnóstico para conseguir objetivos tales como:

- Robustecer y automatizar la red.
- Optimizar la conexión de las zonas con fuentes de energía renovable.
- Desarrollar arquitecturas de generación descentralizadas (generación distribuida).

- Mejorar la integración de la generación intermitente y de nuevas tecnologías de almacenamiento.
- Gestión activa de la demanda.

Uno de los apartados más importantes es el hecho de que al mejorar notablemente la gestión de la demanda local disminuye la redundancia de líneas de transmisión y distribución, aparte de mejorar la eficiencia. Toda esta optimización desembocará en unos ahorros considerables. Además, el robustecimiento y automatización de la red, puede hacer que ésta subsane ella misma errores y fallos como apagones en un tiempo mucho menor.

En lo referente al estado actual, para el 2020 se tiene previsto que en Europa el 80% de los contadores sean de este tipo de dispositivo que debe cumplir, aparte de ciertas características de medida, las siguientes propiedades de autogestión:

- Lectura remota de energía y potencia.
- Lectura remota de parámetros de calidad.
- Programación remota.
- Sincronización remota con equipos concentradores.
- Control remoto de la potencia: Conexión y reconexión a distancia.
- Capacidad de gestión de cargas.

Pero no se deben confundir el tener contadores inteligentes con tener una Smart Grid como hacen la mayoría de los usuarios. Por desgracia el nivel de implantación en España no va mucho más allá y salvo un par de proyectos aislados en Barcelona, Málaga y un estudio en Canarias no se cuenta con ninguna.

Anexo: Apuesta por un Entorno Inteligente Adaptado

En un comienzo me embarqué en un proyecto que trataba la Administración Energética en entornos urbóticos. Por desgracia, según me documentaba más y más el camino a seguir se iba difuminando en un gran abstracto.

Ello se debió principalmente a que en lo referente a Ciudades Inteligentes existen más conceptos que definiciones y ello permite una gran libertad a la hora de orientar una administración energética.

Incluso acudiendo a congresos como el que hubo en Logroño el miércoles 22 de mayo de 2013 sobre Smart Cities no conseguía focalizar la idea.

Así pues, tras un tiempo, decidí volver al origen que me llevó las Smart Cities para acercar mi entorno más a una de ellas sin embarcarme un proyecto que quizás fuese demasiado amplio y abstracto.

De este modo conocí a Embleblue, una empresa que se iniciaba en el internet de las cosas y dispositivos electrónicos de tamaño reducido. Esta empresa podía crear, entre otras cosas, balizas Bluetooth Low Energy. Observando el potencial de dichos elementos para la localización en interiores acabé decidiendo realizar una propuesta de producto basado en sus balizas.

De este modo se podía crear un pequeño entorno inteligente, que a fin de cuentas es un paso más hacia una Ciudad Inteligente.

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

Adaptación de un entorno inteligente mediante balizas BLE para invidentes

PLANOS

Autor: Guillermo García de la Fuente
Tutor: Ignacio R. Matías Maestro
Pamplona, julio 2014

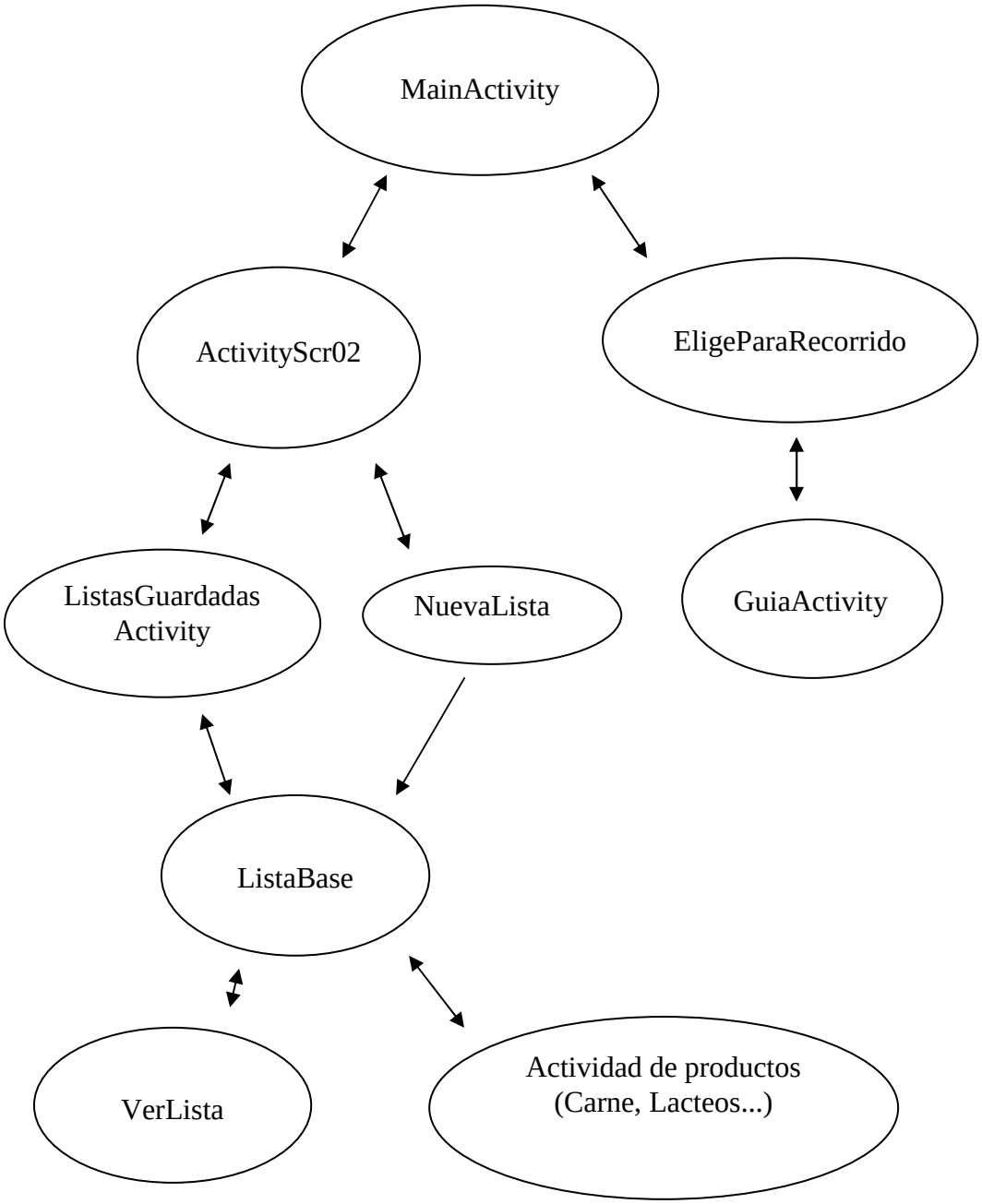
Contenido

1. Introducción

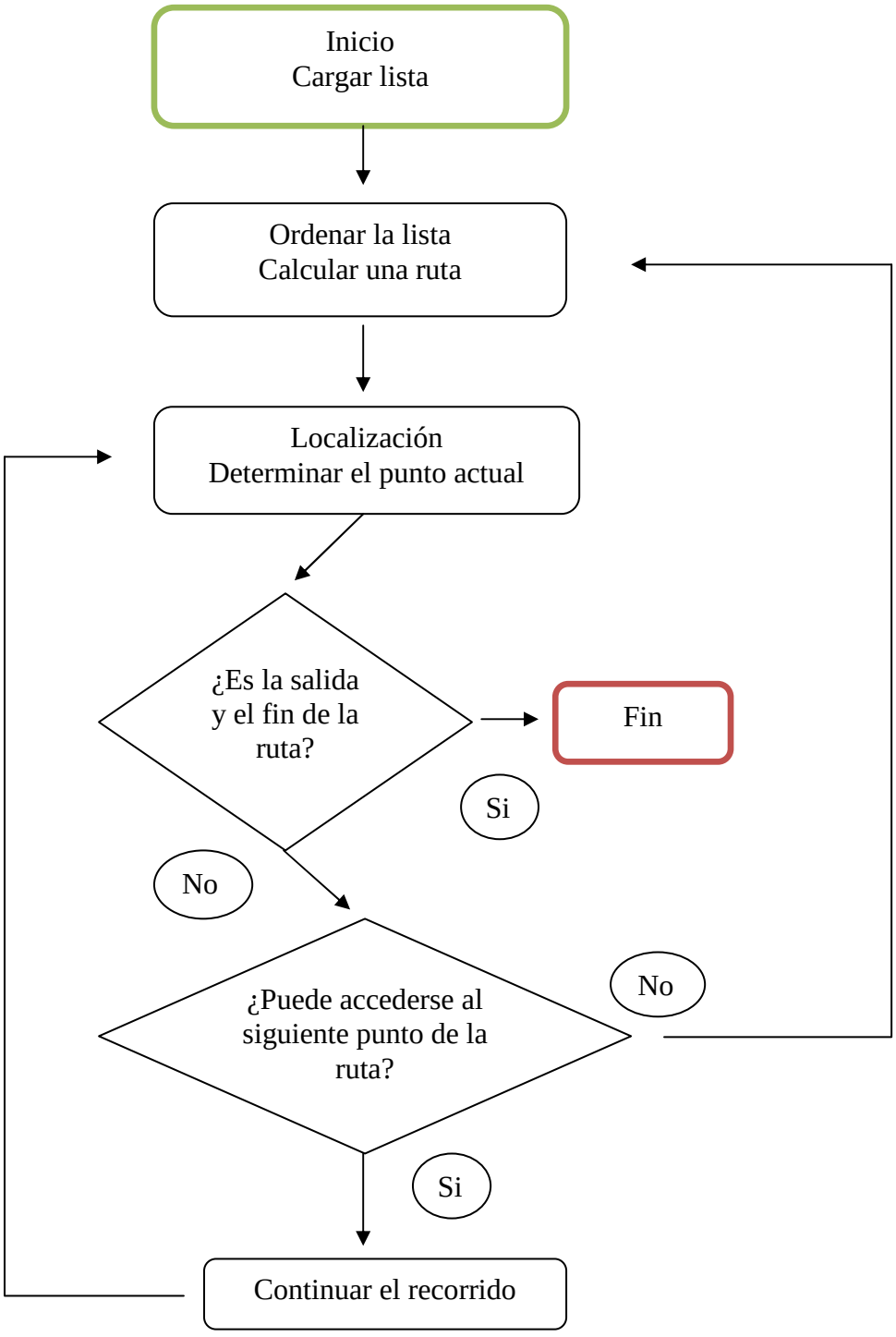
En éste documento se detallará el código utilizado para generar la aplicación. Especialmente las actividades y las clases creadas para dar apoyo a éstas en sus algoritmos.

2. Diagramas de flujo

2.1 Actividades



2.2 Guiado (en GuiaActivity)



3. Código

3.1 Android Manifest

El documento conocido como Android Manifest es en el que la aplicación guarda información de su versión así como permisos especiales que pudiera requerir. En nuestro caso requeríamos permisos para manipular las tecnologías Bluetooth y Wi-Fi.

Es además en éste documento donde han de ser declaradas las actividades utilizadas para que la aplicación pueda utilizarlas:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bshoppingassistant"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="18"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.bshoppingassistant.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.example.bshoppingassistant.ActivityScr02"
>
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.NuevaLista"
            android:label="@string/title_activity_nueva_lista"

android:parentActivityName="com.example.bshoppingassistant.ActivityScr02" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.bshoppingassistant.ActivityScr02"
/>
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.ListaBase"
```

```
            android:label="@string/title_activity_lista_base" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Carne"
            android:label="@string/title_activity_carne" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Pescado"
            android:label="@string/title_activity_pescado" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Lacteos"
            android:label="@string/title_activity_lacteos" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Verduras"
            android:label="@string/title_activity_verduras" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Frutas"
            android:label="@string/title_activity_frutas" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.VerLista"
            android:label="@string/title_activity_ver_lista" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.Ruta"
            android:label="@string/title_activity_ruta" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.MuestraRuta"
            android:label="@string/title_activity_muestra_ruta" >
        </activity>
        <activity

android:name="com.example.bshoppingassistant.ListasGuardadasActivity"
            android:label="@string/title_activity_listas_guardadas" >
        </activity>
        <activity

android:name="com.example.bshoppingassistant.EligeParaRecorridoActivity"
            android:label="@string/title_activity_elige_para_recorrido" >
        </activity>
        <activity
            android:name="com.example.bshoppingassistant.GuiaActivity"
            android:label="@string/title_activity_guia" >
        </activity>
        <service android:name=".BluetoothLeService" android:enabled="true"/>
    </application>

      </manifest>
```

3.2 Actividades

Para el funcionamiento de la aplicación fueron creadas las siguientes actividades:

- MainActivity
- ActivityScr02
- ListasGuardadasActivity
- NuevaLista
- ListaBase
- Carne
- Pescado
- Lacteos
- Verduras
- Frutas
- VerLista
- EligeParaRecorrido
- GuiaActivity

3.2.1 MainActivity

Esta actividad mostraba la pantalla principal de la aplicación dando a elegir entre dos opciones: "Listas guardadas" y "Comenzar compra".

De cada una de ellas se accedería a una nueva actividad. En este caso a ActivityScr02 o EligeParaRecorrido.

```
package com.example.bshoppingassistant;


import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnLongClickListener;
import android.widget.Button;


public class MainActivity extends Activity {

    Button boton1;
    Button boton2;

    TextToSpeech ttobj;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

         boton1 = (Button) findViewById(R.id.button1);
         boton2 = (Button) findViewById(R.id.button2);


         boton1.setOnClickListener(new OnClickListener() {

             @Override
             public void onClick(View v)
             {Intent intent = new Intent(MainActivity.this,
ActivityScr02.class);
                 startActivity(intent);
                 }
         });


         // los listeners para leer el boton
         boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                 public boolean onLongClick(View v) {
                     // TODO Auto-generated method stub
                  String texto = boton1.getText().toString();
                  ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                     return true;
                 }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                 public boolean onLongClick(View v) {
                     // TODO Auto-generated method stub
                  String texto = boton2.getText().toString();
                  ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                     return true;
                 }

         });

    }
```

```
    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


         final Locale locale = new Locale("es", "ES");

            ttobj=new TextToSpeech(getApplicationContext(),
                      new TextToSpeech.OnInitListener() {
                      @Override
                      public void onInit(int status) {
                         if(status != TextToSpeech.ERROR){
                             ttobj.setLanguage(locale);
                             ttobj.setSpeechRate(1.5f);
                            }
                         }
            });
    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }


    public void muestraRuta(View view){
        Intent intent = new Intent(this,
EligeParaRecorridoActivity.class);
        startActivity(intent);
    }

}
```

3.2.2 ActivityScr02

En esta actividad se muestran las opciones para acceder a las listas ya guardadas, crear una nueva lista o volver.

```
package com.example.bshoppingassistant;
```

```
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class ActivityScr02 extends Activity {

    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menulista);

        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);



        boton1.setOnLongClickListener(new OnLongClickListener(){
            @Override
               public boolean onLongClick(View v) {
                   // TODO Auto-generated method stub
                String texto = boton1.getText().toString();
                ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                   return true;
               }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
            @Override
               public boolean onLongClick(View v) {
                   // TODO Auto-generated method stub
                String texto = boton2.getText().toString();
                ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                   return true;
               }

         });


         boton3.setOnLongClickListener(new OnLongClickListener(){
            @Override
               public boolean onLongClick(View v) {
                   // TODO Auto-generated method stub
                String texto = boton3.getText().toString();
```

```
                    ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                       return true;
                    }

            });

    }

    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


        final Locale locale = new Locale("es", "ES");

           ttobj=new TextToSpeech(getApplicationContext(),
                     new TextToSpeech.OnInitListener() {
                     @Override
                     public void onInit(int status) {
                        if(status != TextToSpeech.ERROR){
                            ttobj.setLanguage(locale);
                            ttobj.setSpeechRate(1.5f);
                          }
                        }
           });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }


    public void listasGuardadas(View view){
        Intent intent = new Intent(this,
ListasGuardadasActivity.class);
        startActivity(intent);
    }

    public void crearLista(View view){
        Intent intent = new Intent(this, NuevaLista.class);
        startActivity(intent);
    }

    public void cerrar (View view){
        finish();
    }

}
```

3.2.3 ListasGuardadasActivity

Esta actividad muestra en pantalla una lista mediante el interfaz ListView en la que aparecen las listas creadas por el usuario y son leídas para él. Mediante un clic prolongado el usuario podrá identificar las listas y mediante un solo clic seleccionar la que desea.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class ListasGuardadasActivity extends Activity {

    private final static String noHay = "No hay listas";

    TextToSpeech ttobj;
    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_listas_guardadas);


    }

    @Override
    public void onResume(){
        super.onResume();



        final Locale locale = new Locale("es", "ES");

        ttobj=new TextToSpeech(getApplicationContext(),
                  new TextToSpeech.OnInitListener() {
                  @Override
                  public void onInit(int status) {
                     if(status != TextToSpeech.ERROR){
                         ttobj.setLanguage(locale);
                         ttobj.setSpeechRate(1.5f);
                        }
                     }
        });
```

```
        ListView listView = (ListView)
findViewById(R.id.mListViewVerListas);

        final String indexStr;


        final SharedPreferences pref = getSharedPreferences("listas",
0);
        indexStr = pref.getString("index", noHay); // si no hay listas
o no puede cargar introduce el texto noHay

        final ArrayList<String> indexAlist =
apoyo.stringToAlist(indexStr);

        final ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
android.R.id.text1, indexAlist);

        listView.setAdapter(adapter);


        listView.setOnItemClickListener(
                  new OnItemClickListener() {
                 @Override
                 public void onItemClick(AdapterView<?> parent,
                   View view, int position, long id) {


                     String lista = indexAlist.get(position);
                     if (lista != noHay){
                       abreLista(lista);
                     }



                 }

                });


        listView.setOnItemLongClickListener(
                  new OnItemLongClickListener() {
                 @Override
                 public boolean onItemLongClick(AdapterView<?>
parent,
                   View view, int position, long id) {


                    String voz = indexAlist.get(position);
                    ttobj.speak(voz, TextToSpeech.QUEUE_FLUSH, null);


                  return true;
                 }

                });

        //-------

        mHandler = new Handler();
```

```
        new Thread(new Runnable() {
            @Override
            public void run() {
                    try {
                        Thread.sleep(100);
                        mHandler.post(new Runnable() {

                            @Override
                            public void run() {
                                ttobj.speak(indexStr,
TextToSpeech.QUEUE_FLUSH, null);
                            }
                        });
                    } catch (Exception e) {
                    }

            }
        }).start();



    }

    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.listas_guardadas, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            borraTodas();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    public void abreLista(String lista){
        Intent intent = new Intent(this, ListaBase.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);
    }

    public void borraTodas(){

            SharedPreferences pref = getSharedPreferences("listas",0);
            SharedPreferences.Editor editor = pref.edit();
            editor.remove("index");  //con esto borramos todas las
listas;
            editor.commit();

    }

}
```

3.2.4 NuevaLista

Esta actividad permite la creación de una nueva lista introduciendo un nombre para ella.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.EditText;

public class NuevaLista extends Activity {

    TextToSpeech ttobj;
    private Handler mHandler;
    String introduzca = "Introduzca nombre para la lista";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nueva_lista);


        final Locale locale = new Locale("es", "ES");

        ttobj=new TextToSpeech(getApplicationContext(),
                  new TextToSpeech.OnInitListener() {
                  @Override
                  public void onInit(int status) {
                     if(status != TextToSpeech.ERROR){
                         ttobj.setLanguage(locale);
                         ttobj.setSpeechRate(1.5f);
                        }
                     }
        });



        mHandler = new Handler();


        new Thread(new Runnable() {
            @Override
            public void run() {
                    try {
                        Thread.sleep(100);
                        mHandler.post(new Runnable() {

                            @Override
                            public void run() {
                                ttobj.speak(introduzca,
TextToSpeech.QUEUE_FLUSH, null);
```

```
                        }
                    });
                } catch (Exception e) {
                }

            }
        }).start();



    }

    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }



    public void abreLista(View view){

        final String nulo = "valorNulo";

        final EditText cuadroTexto = (EditText)
findViewById(R.id.nuevoNombre);
        String nuevaLista = cuadroTexto.getText().toString();

        if (!nuevaLista.matches("")){
            SharedPreferences pref = getSharedPreferences("listas",
0);
            String indexStr = pref.getString("index", nulo);
            ArrayList<String> indexList = new ArrayList<String>();

            indexList = apoyo.stringToAlist(indexStr);



            if (!indexList.contains(nuevaLista)){
                indexList.add(nuevaLista);
                if (indexList.contains(nulo)){
                    indexList.remove(indexList.indexOf(nulo));
                }
                indexStr = apoyo.alistToString(indexList);

                SharedPreferences.Editor editor = pref.edit();
                editor.putString("index", indexStr);
                editor.commit();
            }

            Intent intent = new Intent(this, ListaBase.class);
            intent.putExtra("nombreLista", nuevaLista);
```

```
            startActivity(intent);
            finish();
        }
        else {
            ttobj.speak(introduzca, TextToSpeech.QUEUE_FLUSH, null);
        }

    }
}
```

3.2.5 ListaBase

En esta actividad se muestran diversas opciones. En primer lugar las categorías de alimentos y a continuación las opciones para poder ver que elementos se encuentran ya en la lista y la de borrar dicha lista.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class ListaBase extends Activity {

    Button boton1;
    Button boton2;
    Button boton3;
    Button boton4;
    Button boton5;
    Button boton6;
    Button boton7;
    Button boton8;

    TextToSpeech ttobj;



    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista_base);

        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;

        boton1 = (Button) findViewById(R.id.button1);
```

```
boton2 = (Button) findViewById(R.id.button2);
boton3 = (Button) findViewById(R.id.button3);
boton4 = (Button) findViewById(R.id.button4);
boton5 = (Button) findViewById(R.id.button5);
boton6 = (Button) findViewById(R.id.button6);
boton7 = (Button) findViewById(R.id.button7);
boton8 = (Button) findViewById(R.id.button8);


boton1.setOnLongClickListener(new OnLongClickListener(){
    @Override
       public boolean onLongClick(View v) {
           // TODO Auto-generated method stub
        String texto = boton1.getText().toString();
        ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
           return true;
       }

});


boton2.setOnLongClickListener(new OnLongClickListener(){
    @Override
       public boolean onLongClick(View v) {
           // TODO Auto-generated method stub
        String texto = boton2.getText().toString();
        ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
           return true;
       }

});


boton3.setOnLongClickListener(new OnLongClickListener(){
    @Override
       public boolean onLongClick(View v) {
           // TODO Auto-generated method stub
        String texto = boton3.getText().toString();
        ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
           return true;
       }

});


boton4.setOnLongClickListener(new OnLongClickListener(){
    @Override
       public boolean onLongClick(View v) {
           // TODO Auto-generated method stub
        String texto = boton4.getText().toString();
        ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
           return true;
       }

});

boton5.setOnLongClickListener(new OnLongClickListener(){
    @Override
       public boolean onLongClick(View v) {
           // TODO Auto-generated method stub
        String texto = boton5.getText().toString();
```

```
                ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                   return true;
                }

        });

        boton6.setOnLongClickListener(new OnLongClickListener(){
            @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton6.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

        });

        boton7.setOnLongClickListener(new OnLongClickListener(){
            @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton7.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

        });

        boton8.setOnLongClickListener(new OnLongClickListener(){
            @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton8.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

        });



    }


    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


        final Locale locale = new Locale("es", "ES");

           ttobj=new TextToSpeech(getApplicationContext(),
                     new TextToSpeech.OnInitListener() {
                     @Override
                     public void onInit(int status) {
                        if(status != TextToSpeech.ERROR){
                            ttobj.setLanguage(locale);
                            ttobj.setSpeechRate(1.5f);
                           }
```

```
                    }
            });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.lista_base, menu);
        return true;
    }

    public void cerrar (View view){
        finish();
    }

    public void abreCarne(View view){
        Intent intent = new Intent(this, Carne.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }
    public void abrePescado(View view){
        Intent intent = new Intent(this, Pescado.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }
    public void abreVerduras(View view){
        Intent intent = new Intent(this, Verduras.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }
    public void abreFrutas(View view){
        Intent intent = new Intent(this, Frutas.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }
    public void abreLacteos(View view){
        Intent intent = new Intent(this, Lacteos.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }
```

```
    public void verLista(View view){
        Intent intent = new Intent(this, VerLista.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);

    }

    public void borrarLista(View view){
        SharedPreferences pref = getSharedPreferences("listas",0);
        SharedPreferences.Editor editor = pref.edit();
        editor.remove(lista);

        String noHay = "No hay listas";
        String indexStr = pref.getString("index", noHay);
        ArrayList<String> indexList = apoyo.stringToAlist(indexStr);

        if (indexList.contains(lista)){
            indexList.remove(indexList.indexOf(lista));
        }

        if (!indexList.isEmpty()){
            indexStr = apoyo.alistToString(indexList);
        }
        else {
            indexStr = noHay;
        }

        editor.putString("index", indexStr);

        editor.commit();
        finish();
    }

}
```

3.2.6 Carne

Esta actividad corresponde con una categoría de alimentos en la que se encuentran dos productos. Las siguientes actividades son muy similares.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class Carne extends Activity {
```

```
    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;


    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_carne);


        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;


        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);


        boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                 String texto = boton1.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                 String texto = boton2.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });

         boton3.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                 String texto = boton3.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });

    }


    @Override
```

```
protected void onResume() {
    super.onResume();

  //Preparamos el poder leer los botones (poner en Onresume)


     final Locale locale = new Locale("es", "ES");

        ttobj=new TextToSpeech(getApplicationContext(),
                  new TextToSpeech.OnInitListener() {
                  @Override
                  public void onInit(int status) {
                     if(status != TextToSpeech.ERROR){
                         ttobj.setLanguage(locale);
                         ttobj.setSpeechRate(1.5f);
                        }
                     }
        });


}


@Override
public void onPause(){

    //Finalizamos el lector de botones
    super.onPause();
    if(ttobj !=null){
        ttobj.stop();
        ttobj.shutdown();
     }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.carne, menu);
    return true;
}


public void cerrar (View view){
    finish();
}

public void anadeTernera(View view){

    String texto = "Ternera";
    SharedPreferences pref = getSharedPreferences("listas", 0);
    String listaStr = pref.getString(lista, "valorNulo");
    ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

    if (!listaAlist.contains(texto)){
        listaAlist.add(texto);
        if (listaAlist.contains("valorNulo")){
            listaAlist.remove(listaAlist.indexOf("valorNulo"));
        }
        listaStr = apoyo.alistToString(listaAlist);
        SharedPreferences.Editor editor = pref.edit();
        editor.putString(lista, listaStr);
        editor.commit();
```

```
        }

    }


    public void anadePollo(View view){

        String texto = "Pollo";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }

}
```

3.2.7 Pescado

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class Pescado extends Activity {

    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;


    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pescado);


        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;


        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);


        boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton1.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton2.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton3.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton3.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });

    }



    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


         final Locale locale = new Locale("es", "ES");
```

```
            ttobj=new TextToSpeech(getApplicationContext(),
                      new TextToSpeech.OnInitListener() {
                      @Override
                      public void onInit(int status) {
                         if(status != TextToSpeech.ERROR){
                             ttobj.setLanguage(locale);
                             ttobj.setSpeechRate(1.5f);
                            }
                         }
            });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.pescado, menu);
        return true;
    }

    public void cerrar (View view){
        finish();
    }

    public void anadeSardinas(View view){

        String texto = "Sardinas";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }
    }


    public void anadeSalmon(View view){
```

```
        String texto = "Salmon";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }

}
```

3.2.8 Lacteos

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class Lacteos extends Activity {


    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;



    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lacteos);


        Intent intent = getIntent();
```

```
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;



        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);



        boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton1.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton2.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton3.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton3.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });



    }


    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


         final Locale locale = new Locale("es", "ES");

            ttobj=new TextToSpeech(getApplicationContext(),
```

```
                    new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                       if(status != TextToSpeech.ERROR){
                           ttobj.setLanguage(locale);
                           ttobj.setSpeechRate(1.5f);
                          }
                       }
            });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.lacteos, menu);
        return true;
    }

    public void cerrar (View view){
        finish();
    }

    public void anadeYogurt(View view){

        String texto = "Yogurt";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }


    public void anadeLeche(View view){
```

```
        String texto = "Leche";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }

}
```

3.2.9 Verduras

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class Verduras extends Activity {


    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;

    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_verduras);

        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;
```

```
        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);


        boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton1.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton2.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton3.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton3.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


    }


    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


         final Locale locale = new Locale("es", "ES");

            ttobj=new TextToSpeech(getApplicationContext(),
                      new TextToSpeech.OnInitListener() {
                      @Override
                      public void onInit(int status) {
                         if(status != TextToSpeech.ERROR){
                             ttobj.setLanguage(locale);
```

```
                            ttobj.setSpeechRate(1.5f);
                        }
                    }
            });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }



    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.verduras, menu);
        return true;
    }

    public void cerrar (View view){
        finish();
    }


    public void anadeZanahoria(View view){

        String texto = "Zanahoria";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }


    public void anadePuerro(View view){

        String texto = "Puerro";
        SharedPreferences pref = getSharedPreferences("listas", 0);
```

```
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }

}
```

3.2.10 Frutas

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.view.View.OnLongClickListener;
import android.widget.Button;

public class Frutas extends Activity {


    Button boton1;
    Button boton2;
    Button boton3;


    TextToSpeech ttobj;

    private static String lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_frutas);



        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;
```

```
        boton1 = (Button) findViewById(R.id.button1);
        boton2 = (Button) findViewById(R.id.button2);
        boton3 = (Button) findViewById(R.id.button3);


        boton1.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton1.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton2.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton2.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });


         boton3.setOnLongClickListener(new OnLongClickListener(){
             @Override
                public boolean onLongClick(View v) {
                    // TODO Auto-generated method stub
                 String texto = boton3.getText().toString();
                 ttobj.speak(texto, TextToSpeech.QUEUE_FLUSH, null);
                    return true;
                }

         });

    }


    @Override
    protected void onResume() {
        super.onResume();

      //Preparamos el poder leer los botones (poner en Onresume)


         final Locale locale = new Locale("es", "ES");

            ttobj=new TextToSpeech(getApplicationContext(),
                      new TextToSpeech.OnInitListener() {
                      @Override
                      public void onInit(int status) {
                         if(status != TextToSpeech.ERROR){
                             ttobj.setLanguage(locale);
                             ttobj.setSpeechRate(1.5f);
```

```
                        }
                    }
            });


    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.frutas, menu);
        return true;
    }

    public void cerrar (View view){
        finish();
    }

    public void anadeManzana(View view){

        String texto = "Manzana";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);

        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }


    public void anadePlatano(View view){

        String texto = "Platano";
        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(lista, "valorNulo");
        ArrayList<String> listaAlist = apoyo.stringToAlist(listaStr);
```

```
        if (!listaAlist.contains(texto)){
            listaAlist.add(texto);
            if (listaAlist.contains("valorNulo")){
                listaAlist.remove(listaAlist.indexOf("valorNulo"));
            }
            listaStr = apoyo.alistToString(listaAlist);
            SharedPreferences.Editor editor = pref.edit();
            editor.putString(lista, listaStr);
            editor.commit();

        }

    }

}
```

3.2.11 VerLista

Esta actividad al ejecutarse preparará un texto con los elementos de la lista y la leerá al usuario. Desde ella mediante un clic prolongado el usuario podrá identificar sobre que elemento se encuentra y con un clic simple podrá borrarlo si lo desea.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class VerLista extends Activity {


    TextToSpeech ttobj;
    private Handler mHandler;

    private static String lista;
    private static final String vacia = "Lista vac;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ver_lista);

        ListView listView = (ListView)
findViewById(R.id.mListViewVerLista);

        Intent intent = getIntent();
```

```
        String listaHeredada = intent.getStringExtra("nombreLista");
        lista = listaHeredada;


            //le damos el mensaje desde preferencias
            SharedPreferences pref = getSharedPreferences("listas",0);
            String texto = pref.getString(lista, vacia);


            if (texto != vacia && texto != ""){
                texto = Ordenador.ordena(texto);
                SharedPreferences.Editor editor = pref.edit();
                editor.putString(lista, texto);
                editor.commit();
                }

            final ArrayList<String> aList =
apoyo.stringToAlist(texto);

            if (aList.contains("Entrada")){
                int borrar =aList.indexOf("Entrada");
                aList.remove(borrar);}

            if (aList.contains("Salida")){
                int borrar =aList.indexOf("Salida");
                aList.remove(borrar);
            }

            if (aList.isEmpty()){
                aList.add(vacia);
            }

            final ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
android.R.id.text1, aList);

            listView.setAdapter(adapter);


            final Locale locale = new Locale("es", "ES");

            ttobj=new TextToSpeech(getApplicationContext(),
                      new TextToSpeech.OnInitListener() {
                      @Override
                      public void onInit(int status) {
                         if(status != TextToSpeech.ERROR){
                             ttobj.setLanguage(locale);
                             ttobj.setSpeechRate(1.5f);
                            }
                         }
            });


            final String voz = apoyo.alistToString(aList);


            //hacemos que se ejecute algo m□tarde porque si no no
funciona
            mHandler = new Handler();

            new Thread(new Runnable() {
```

```
                @Override
                public void run() {
                        try {
                            Thread.sleep(100);
                            mHandler.post(new Runnable() {

                                @Override
                                public void run() {
                                    ttobj.speak(voz,
TextToSpeech.QUEUE_FLUSH, null);
                                }
                            });
                        } catch (Exception e) {
                        }

                }
            }).start();


            listView.setOnItemClickListener(
                       new OnItemClickListener() {
                      @Override
                      public void onItemClick(AdapterView<?> parent,
                        View view, int position, long id) {

                          //comienza el m鷑do
                          if (!aList.isEmpty()){
                              aList.remove(position);



                          ArrayList<String> listAux = new
ArrayList<String>();


                          if (!aList.isEmpty()){
                              String voz =
apoyo.alistToString(aList);
                              ttobj.speak(voz,
TextToSpeech.QUEUE_FLUSH, null);

                                listAux.add("Entrada");
                                listAux.addAll(aList);
                                listAux.add("Salida");
                          }

                           if (aList.isEmpty()){
                              aList.add(vacia);
                              ttobj.speak(vacia,
TextToSpeech.QUEUE_FLUSH, null);
                              listAux.clear();
                          }

                            SharedPreferences pref =
getSharedPreferences("listas",0);
                            SharedPreferences.Editor editor =
pref.edit();
                            editor.putString(lista,
apoyo.alistToString(listAux));
                            if (aList.contains(vacia))
editor.remove(lista);
```

```
                                editor.commit();
                            }
                            else {
                               ttobj.speak(vacia,
TextToSpeech.QUEUE_FLUSH, null);
                            }
                           adapter.notifyDataSetChanged();

                            //acaba el m鮗do

                        }
                       });


            listView.setOnItemLongClickListener(
                   new OnItemLongClickListener() {
                  @Override
                  public boolean onItemLongClick(AdapterView<?>
parent,
                   View view, int position, long id) {

                      //comienza el m鮗do realmente

                      String voz = aList.get(position);
                     ttobj.speak(voz, TextToSpeech.QUEUE_FLUSH, null);

                     //fin del m鮗do

                          return true;
                   }

                 });
    }


    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
        finish();
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.ver_lista, menu);
        return true;
    }

}
```

3.2.12 EligeParaRecorrido

En esta actividad, al igual que en la que permitía acceder a una lista para modificarla se muestran las listas disponibles y son leídas para el usuario. Una vez elegida una se procederá a la actividad de guiado.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Locale;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class EligeParaRecorridoActivity extends Activity {


    TextToSpeech ttobj;
    private Handler mHandler;


    private final static String noHay = "No hay listas";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_elige_para_recorrido);


    }

    @Override
    public void onResume(){
        super.onResume();


        final Locale locale = new Locale("es", "ES");

        ttobj=new TextToSpeech(getApplicationContext(),
                  new TextToSpeech.OnInitListener() {
                  @Override
                  public void onInit(int status) {
                     if(status != TextToSpeech.ERROR){
                         ttobj.setLanguage(locale);
                         ttobj.setSpeechRate(1.5f);
                        }
                     }
        });
```

```
        ListView listView = (ListView)
findViewById(R.id.mListViewVerListas);

        final String indexStr;


        final SharedPreferences pref = getSharedPreferences("listas",
0);
        indexStr = pref.getString("index", noHay);

        final ArrayList<String> indexAlist =
apoyo.stringToAlist(indexStr);

        final ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
android.R.id.text1, indexAlist);

        listView.setAdapter(adapter);


        listView.setOnItemClickListener(
                  new OnItemClickListener() {
                 @Override
                 public void onItemClick(AdapterView<?> parent,
                   View view, int position, long id) {


                     String lista = indexAlist.get(position);
                     if (lista != noHay){
                       abreLista(lista);
                     }



                 }

                });


        listView.setOnItemLongClickListener(
                  new OnItemLongClickListener() {
                 @Override
                 public boolean onItemLongClick(AdapterView<?>
parent,
                  View view, int position, long id) {

                     //comienza el m鮗do realmente

                     String voz = indexAlist.get(position);
                    ttobj.speak(voz, TextToSpeech.QUEUE_FLUSH, null);

                    //fin del m鮗do

                         return true;
                 }

                });
```

```
        mHandler = new Handler();


        new Thread(new Runnable() {
            @Override
            public void run() {
                    try {
                        Thread.sleep(100);
                        mHandler.post(new Runnable() {

                            @Override
                            public void run() {
                                ttobj.speak(indexStr,
TextToSpeech.QUEUE_FLUSH, null);
                            }
                        });
                    } catch (Exception e) {
                    }

            }
        }).start();


    }

    @Override
    public void onPause(){

        //Finalizamos el lector de botones
        super.onPause();
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
        finish();
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it
is present.
        getMenuInflater().inflate(R.menu.elige_para_recorrido, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    public void abreLista(String lista){
        Intent intent = new Intent(this, GuiaActivity.class);
        intent.putExtra("nombreLista", lista);
        startActivity(intent);
    }


}
```

3.2.13 GuiaActivity

Esta actividad será la encargada de guiar al usuario por el recorrido. Cada vez que el usuario se salga del recorrido volverá a calcular la ruta óptima desde su nueva posición. Además dará indicaciones verbales sobre la dirección a seguir.

Cada 3 segundos ejecuta el escaneo de dispositivos durante un peridodo de tiempo de 2 segundos. De este modo no consume siempre batería y permite una buena detección de los dispositivos cercanos.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Locale;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Handler;
import android.speech.tts.TextToSpeech;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

public class GuiaActivity extends Activity {


        private BluetoothAdapter mBluetoothAdapter;
        private boolean mScanning;
        private Handler mHandler;


        private static final int REQUEST_ENABLE_BT = 1;
        // Stops scanning after 1.5 seconds.
        private static final long SCAN_PERIOD = 2000;
        private static final long SCAN_REPEAT = 3000;

      //para el paseo
```

```
        private static ArrayList<String> listaAlist = new
ArrayList<String>();
        private static ArrayList<Integer> recorrido = new
ArrayList<Integer>();

        TextToSpeech ttobj;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_guia);


        getActionBar().setTitle(R.string.guia);
        mHandler = new Handler();

        //inicializaci□T

        if
(!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOT
H_LE)) {
            Toast.makeText(this, R.string.ble_not_supported,
Toast.LENGTH_SHORT).show();
            finish();
        }

        final BluetoothManager bluetoothManager =
                (BluetoothManager)
getSystemService(Context.BLUETOOTH_SERVICE);
        mBluetoothAdapter = bluetoothManager.getAdapter();

        if (mBluetoothAdapter == null) {
            Toast.makeText(this,
R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }

        //inicializa el mapa de puntos para vecinos, etc
        Buscamino.preparaMapa();
        Tablas.iniciar();
        Posiciones.lugar.put("actual",
Posiciones.lugar.get("Entrada"));

        //cargamos el nombre de la lista elegida
        Intent intent = getIntent();
        String listaHeredada = intent.getStringExtra("nombreLista");

        SharedPreferences pref = getSharedPreferences("listas", 0);
        String listaStr = pref.getString(listaHeredada, "valorNulo");
        listaAlist = apoyo.stringToAlist(listaStr);

        //eliminamos la entrada de la lista en caso de que est錬
if (listaAlist.contains("Entrada")){
            listaAlist.remove(listaAlist.indexOf("Entrada"));
        }

        recalculando();
```

```
        AdminLista.actualiza();



        TextView txtPosicion = (TextView)
findViewById(R.id.posicionActual);
        txtPosicion.setText("Localizando");




        final Locale locale = new Locale("es", "ES");

        ttobj=new TextToSpeech(getApplicationContext(),
                  new TextToSpeech.OnInitListener() {
                  @Override
                  public void onInit(int status) {
                     if(status != TextToSpeech.ERROR){
                         ttobj.setLanguage(locale);
                         ttobj.setSpeechRate(1.5f);
                        }
                     }
        });


        //inicializar la posici□n la entrada
        Posiciones.lugar.put("actual",
Posiciones.lugar.get("Entrada"));

        //inicializar la direcci□n la que cree mirar
        Direccion.iniciaAb();


    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {


        getMenuInflater().inflate(R.menu.guia, menu);
        if (!mScanning) {
            menu.findItem(R.id.menu_stop).setVisible(false);
            menu.findItem(R.id.menu_scan).setVisible(true);
            menu.findItem(R.id.menu_refresh).setActionView(null);
        } else {
            menu.findItem(R.id.menu_stop).setVisible(true);
            menu.findItem(R.id.menu_scan).setVisible(false);
            menu.findItem(R.id.menu_refresh).setActionView(
                    R.layout.actionbar_indeterminate_progress);
        }
        return true;

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        switch (item.getItemId()) {
            case R.id.menu_scan:
                //mLeDeviceListAdapter.clear();
```

```
                scanLeDevice(true);
                break;
            case R.id.menu_stop:
                scanLeDevice(false);
                break;
        }
        return true;
        }

    @Override
    public void onPause(){
        super.onPause();
        scanLeDevice(false);
        if(ttobj !=null){
            ttobj.stop();
            ttobj.shutdown();
         }
        finish();
    }

    @Override
    protected void onResume() {
        super.onResume();

        // Ensures Bluetooth is enabled on the device.  If Bluetooth
is not currently enabled,
        // fire an intent to display a dialog asking the user to grant
permission to enable it.
        if (!mBluetoothAdapter.isEnabled()) {
            if (!mBluetoothAdapter.isEnabled()) {
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent,
REQUEST_ENABLE_BT);
            }
        }

        // Initializes list view adapter.
        //scanLeDevice(true);
        bucleScan();
    }

    private void bucleScan(){

        scanLeDevice(true); //lo inicia una vez

        new Thread(new Runnable() {
            @Override
            public void run() {
                while (!recorrido.isEmpty()) {
                    try {
                        Thread.sleep(SCAN_REPEAT);
                        mHandler.post(new Runnable() {

                            @Override
                            public void run() {
                                scanLeDevice(true);
                            }
                        });
                    } catch (Exception e) {
                    }
```

```
                }

                //TODO fin de compra al salir del while aqui.

                scanLeDevice(false);

                if(ttobj !=null){
                    ttobj.stop();
                    ttobj.shutdown();
                 }

                finish();
            }
        }).start();


    }

    private void scanLeDevice(final boolean enable) {
        if (enable) {
            // Stops scanning after a pre-defined scan period.
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {

                    //TODO completar algoritmo de "paseo"

                    mScanning = false;
                    mBluetoothAdapter.stopLeScan(mLeScanCallback);

                    //aquntroducimos la secuencia para localizarnos
                    int[] vectorRssi = Tablas.getVectorMedia();
                    int idActual =
Buscamino.queId(Posiciones.lugar.get("actual"));
                    int idLocal = Localizacion.localiza(idActual,
vectorRssi);
                    TextView txtPosicion = (TextView)
findViewById(R.id.posicionActual);
                    txtPosicion.setText("ID: "+ idLocal);

                    Direccion.setDireccion(idActual, idLocal); //an no
ha actualiado actual, nos da la direccion que hemos seguido

                    Posiciones.lugar.put("actual",
Buscamino.buscaID(idLocal));




                    if (!recorrido.isEmpty()){
                        if (!Buscamino.esVecinoId(idLocal,
recorrido.get(0))){
                            avanza(idLocal); //en el ltimo instante
vacrecorrido, precauci                                }


                        if (!recorrido.isEmpty()){
```

```
                            TextView txtlista = (TextView)
findViewById(R.id.cosaslista);
                            ArrayList<String> auxiliar = new
ArrayList<String>();
                            auxiliar.addAll(listaAlist);

                            if (auxiliar.contains("actual")){
                                int borrar
=auxiliar.indexOf("actual");
                                auxiliar.remove(borrar);}
                            if (auxiliar.contains("Salida")){
                                int borrar
=auxiliar.indexOf("Salida");
                                auxiliar.remove(borrar);}


txtlista.setText(apoyo.alistToString(auxiliar));

                            if (!auxiliar.isEmpty()){
                                if (idLocal ==
AdminLista.dimeId(auxiliar.get(0))){
                                    String recoger = "Recoja ";

                                    for (int i=0; i<auxiliar.size();
i++){
                                        if (idLocal ==
AdminLista.dimeId(auxiliar.get(i))) recoger += auxiliar.get(i) + ", ";
                                    }
                                    ttobj.speak(recoger,
TextToSpeech.QUEUE_FLUSH, null);

                                }
                            }


                            AdminLista.checkProduct(idLocal);

                            String direccionAseguir =
Direccion.direccionHacia(idLocal, recorrido.get(0));
                            String indicaciones =
Indicacion.getIndicacion(Direccion.actual(), direccionAseguir);

                            TextView txtIndicacion = (TextView)
findViewById(R.id.indicaciones);

                            if (indicaciones !=
((TextView)txtIndicacion).getText().toString()){
                                ttobj.speak(indicaciones,
TextToSpeech.QUEUE_ADD, null); //lee las indicaciones
                            }

                            txtIndicacion.setText(indicaciones);


                            //para mostrar los puntos que quedan
                            String textoRec = "";
                            for (int i=0; i<recorrido.size()-1; i++){
                                textoRec += recorrido.get(i) + ", ";
                            }
                            textoRec +=
recorrido.get(recorrido.size()-1);
```

```
                            TextView txtrec = (TextView)
findViewById(R.id.recorrer);
                            txtrec.setText(textoRec);

                        }
                    }


                    invalidateOptionsMenu();



                }
            }, SCAN_PERIOD);


            //a□do
            Tablas.limpiar();

            mScanning = true;
            mBluetoothAdapter.startLeScan(mLeScanCallback);
        } else {
            mScanning = false;
            mBluetoothAdapter.stopLeScan(mLeScanCallback);
        }
        invalidateOptionsMenu();
    }



    // Device scan callback.
    private BluetoothAdapter.LeScanCallback mLeScanCallback =
            new BluetoothAdapter.LeScanCallback() {

        @Override
        public void onLeScan(final BluetoothDevice device, final int
rssi, byte[] scanRecord) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {

                        int potInt = rssi;
                        if (potInt==0) potInt = -99;
                        String direccion = device.getAddress();

                        if (Tablas.valores.containsKey(direccion)){
                            Tablas.anade(direccion, potInt);
                        }

                    WifiManager myWifiManager =
(WifiManager)getSystemService(Context.WIFI_SERVICE);

                    WifiInfo myWifiInfo =
myWifiManager.getConnectionInfo();
                    int rssiWifi = myWifiInfo.getRssi();
                    if (rssiWifi == 0) rssiWifi = -99;

                    Tablas.anade(Tablas.localWifi, rssiWifi);
                }
            });
        }
```

```
    };


    public static void recalculando(){
        if (!listaAlist.isEmpty()){
            String lista = apoyo.alistToString(listaAlist);
            lista = Ordenador.ordena(lista, "actual");
            listaAlist = apoyo.stringToAlist(lista);
            recorrido = Ruta.creaRuta(lista);
            recorrido.remove(0);
            AdminLista.actualiza();
        }
    }

    public static void avanza(int idActual){
        if (recorrido.get(0) == idActual){
            recorrido.remove(0);
        }
        else{
            recalculando();
        }

    }


    public static class Indicacion{

        private static String ar = "Arriba";
        private static String ab = "Abajo";
        private static String izq = "Izquierda";
        private static String dcha = "Derecha";

        private static String recto = "Siga recto";
        private static String gIzq = "Gire a su izquierda y avance";
        private static String gDcha = "Gire a su derecha y avance";
        private static String vuelta = "De media vuelta y avance";

        private static final HashMap<String, int[]> direcciones;
        static
        {

            direcciones = new HashMap<String, int[]>();

            int[] arriba = {-1, 0};
            direcciones.put(ar, arriba);

            int[] abajo = {1, 0};
            direcciones.put(ab ,abajo);

            int[] izquierda = {0, -1};
            direcciones.put(izq ,izquierda);

            int[] derecha = {0, 1};
            direcciones.put(dcha ,derecha);
        }


        public static String getIndicacion(String dirActual, String
dirNueva){

            if (dirActual == dirNueva) return recto;
```

```
            int[] actual = direcciones.get(dirActual);
            int[] nueva = direcciones.get(dirNueva);
            int resultado = actual[0]*nueva[1] - actual[1]*nueva[0];

            if (resultado>0) return gIzq;
            else if (resultado<0) return gDcha;
            else return vuelta;

        }

    }



    public static class Direccion{

        private static String ar = "Arriba";
        private static String ab = "Abajo";
        private static String izq = "Izquierda";
        private static String dcha = "Derecha";


        private static String direccionActual = ab;

        public static void iniciaAb(){
            direccionActual = ab;
        }

        public static void setDireccion(int idOrigen, int idDestino){
            direccionActual = direccionHacia(idOrigen, idDestino);
        }

        public static String direccionHacia(int idOrigen, int
idDestino){

            String devolver = direccionActual;

            if (idOrigen!=idDestino){
                int [] origen = Buscamino.buscaID(idOrigen);
                int [] destino = Buscamino.buscaID(idDestino);
                int[] resultado = new int[2];


                for (int i = 0; i<resultado.length; i++){
                    resultado[i] = destino[i]-origen[i];
                }

                if (resultado[1]==0){
                    if (resultado[0]>0){
                        devolver = ab;
                    }
                    else{
                        devolver = ar;
                    }
                }
                else{
                    if (resultado[1]>0){
                        devolver = dcha;
                    }
```

```
                    else{
                        devolver = izq;
                    }
                }

            }
            return devolver;

        }

        public static String actual(){
            return direccionActual;
        }
    }

    public static class AdminLista{
        private static final String salida = "Salida";
        private static int idNext;

        public static void actualiza(){

            setIdNext(dimeId(listaAlist.get(1)));

        }

        private static void setIdNext(int id){
            idNext = id;
        }

        public static int dimeId(String producto){
            int[] posicion = Posiciones.lugar.get(producto);
            int id = Buscamino.queId(posicion);
            return id;
        }


        public static void checkProduct(int idActual){
            if (!listaAlist.isEmpty()){
                if (idActual == idNext && listaAlist.get(1) !=
salida){

                    while (idActual == dimeId(listaAlist.get(1)) &&
listaAlist.get(1) != salida){
                        listaAlist.remove(1);
                        actualiza();
                    }

                }
            }

        }

        public static boolean finCompra(){
            int idActual = dimeId("actual");
            if (idActual == dimeId(salida) && listaAlist.get(0) ==
salida) return true;
            else return false;
        }

    }
}
```

3.3 Clases

Las clases fueron implementadas principalmente como lugares donde guardar métodos de uso frecuente o algunos valores estáticos que serían necesarios durante los procesos de guiado, guardado de la información, etc.

Se utilizaron las siguientes:

- apollo
- Buscamino
- Localización
- Ordenador
- Posiciones
- Punto
- Ruta
- Tablas

3.3.1 apollo

En esta clase se guardaron algunos métodos básicos de acceso frecuente desde varias actividades. Estos métodos llevaban a cabo tareas sencillas como la conversión de un String a un ArrayList y viceversa.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Arrays;

public class apoyo {


    public static String alistToString(ArrayList<String> entrada){
        return arrayToString(alistToArray(entrada));
    }

    public static String[] alistToArray(ArrayList<String> entrada){
        String[] salida = new String[entrada.size()];
        salida = entrada.toArray(salida);
        return salida;
    }

    public static ArrayList<String> stringToAlist(String text){
        ArrayList<String> salida = new
ArrayList<String>(Arrays.asList(text.split(", ")));
        return salida;
```

```
    }

    public static String arrayToString(String[] array){
        String salida = Arrays.toString(array);
        salida = salida.replace("[","").replace("]","");
        return salida;
    }
}
```

3.3.2 Buscamino

Sin duda la clase que más trabajo requirió. En ella se almacenan los métodos para definir claramente el mapa, asignar IDs a las posiciones posibles, definir sus vecino y localizar una ruta relativamente corta entre ellos. Lo realiza mediante el algoritmo A* en el método "buscaCamino" dadas unas coordenadas X e Y de inicio y fin.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;


public class Buscamino {
    // los 1 son camino, 0 pared
    static int[][] map = Posiciones.mapa; //apunta al mapa dado en
"Posiciones"



    static int puntero;
    static int filas = map.length;
    static int columnas = map[0].length;

    static int[][] mapId = new int[filas][columnas];

    static Punto[][] mapPuntos = new Punto[filas][columnas];

    static Punto inicio;
    static Punto destino;

    static ArrayList<Punto> open = new ArrayList<Punto>();
    static ArrayList<Punto> closed = new ArrayList<Punto>();
    static ArrayList<Punto> camino = new ArrayList<Punto>();


    //-----------------------------------------------------------------
-------------------------

    public static ArrayList<Punto> buscaCamino(int iniX, int iniY, int
destX, int destY){


        Punto siguiente;
        Punto actual;
        Punto auxiliar = new Punto();

        preparaMapa();
```

```
        //inicializaci□e los puntos mediante la entrada

        inicio = mapPuntos[iniX][iniY];

        destino = mapPuntos[destX][destY];
        destino.g=0;
        destino.f=0;

        //a□mos el punto de inicio a la lista open

        inicio.g = 0;
        inicio.h = valorH(inicio, destino);
        inicio.f = inicio.g + inicio.h;



        //inicializaci□
        closed.clear();
        open.clear();
        camino.clear();

        closed.add(inicio);
        anadeVecinos(inicio);



        if (open.size()>0 && inicio!=destino){

            do{
                if (open.isEmpty()==true){
                    //System.out.println("Se agotaron las
posibilidades.");
                    break;
                }

                puntero = iMenorF();
                auxiliar = open.get(puntero);

                anadeVecinos(auxiliar);
                closed.add(auxiliar);
                open.remove(puntero);



            } while (open.contains(destino)==false);


        }


        closed.add(destino);
        destino.g=auxiliar.g+1;



            actual = destino;
            siguiente = auxiliar; //por inicializar
```

```
            do{
                for (int i=(closed.indexOf(actual) - 1); i>=0; i--){
                        siguiente =  closed.get(i);
                        if (esVecino(siguiente, actual) &&
siguiente.g<actual.g){
                            camino.add(actual);
                            actual = siguiente;
                            break;
                        }
                }


            }while(actual.iD!=inicio.iD);

            camino.add(inicio);


            Collections.reverse(camino); //estaba al reves, lo ponemos
bien

            return camino;
    }

    //-----------------------------------------------------------------
-------------------



    public static void preparaMapa(){
        rellenaMapP();
        setAllVecinos();
    }

    //imprime las ids de los puntos a recorrer

    public static void muestraCamino(){

        Punto punto;

        System.out.println("El camino a recorrer es:");


        for (int i=camino.size()-1; i>=0; i--){
            punto=camino.get(i);
            System.out.println(""+punto.iD);
        }
    }

    //imprime el array del mapa
    public static void muestraMapa(){
        for (int i=0; i<filas; i++){
            System.out.println(Arrays.toString(map[i]));
        }
    }

    //imprime un array con las distintas id

    public static void muestraId(){
        for (int i=0; i<filas; i++){
            System.out.println(Arrays.toString(mapId[i]));
        }
```

```
}


//comprueba si el punto A es vecino del punto B

public static boolean esVecino(Punto puntoA, Punto puntoB){
    for (int i=0; i<puntoB.vecinosPasables.length; i++){
        if (puntoB.vecinosPasables[i]==0) break;
        if (puntoA.iD==puntoB.vecinosPasables[i]) return true;
    }
    return false;
}

public static boolean esVecinoId(int idA, int idB){
    Punto puntoA = buscaPuntoId(idA);
    Punto puntoB = buscaPuntoId(idB);

    return esVecino(puntoA, puntoB);
}


// a□ vecinos pasables del punto a tratar en la lista open

public static void anadeVecinos(Punto punto){

    Punto vecino;
    int id;
    int g;
    int h;
    int f;
    int[] posicion = new int[2];


    for (int i=0; i < punto.vecinosPasables.length; i++)
        if (punto.vecinosPasables[i] == 0) break;
        else {
            id = punto.vecinosPasables[i];
            posicion = buscaID(id);
            vecino = mapPuntos[posicion[0]][posicion[1]];

            if (closed.contains(vecino)==false){

                g= punto.g+1;
                h= valorH(vecino, destino);
                f= g+h;

                if (g<vecino.g || vecino.g==-1){
                    vecino.g = g;
                    vecino.h = h;
                    vecino.f = f;
                }
                if (open.contains(vecino)==false){
                    open.add(vecino);
                }
            }
        }


}
```

```
    // local el ice del punto con menor F

    public static int iMenorF(){
        Punto punto;
        punto = open.get(0);
        int aux = punto.f;
        int puntero = 0;

        if (open.size() > 1){
            for (int i=1; i<open.size(); i++){
                punto = open.get(i);
                if (punto.f < aux){
                    aux = punto.f;
                    puntero = i;
                }

            }
        }

        return puntero;
    }



    // coge la informaci□el mapa array y construye uno con los Puntos
    // la ID empieza en 1 siendo 0 reservado para puntos fuera del
mapa o no validos
    // tambi鮠rellena mapId

    public static void rellenaMapP(){

        int indice = 10; //empezar en 1, puesto en 10 para ver mejor
los array al mostrarlos en pantalla.

        for (int i=0; i<filas; i++){
            for (int k = 0; k<columnas; k++){
                mapPuntos[i][k] = new Punto(indice,map[i][k], i, k);
                mapId[i][k] = indice;
                indice++;
            }
        }
    }


    //Localiza una id en mapPuntos y devuelve su posici·    public
static int[] buscaID(int id){
        int[] posicion = new int[2];
        boolean encontrado = false;

        for (int i=0; i<filas-1; i++){
            if (i+1 <= filas){
                if (mapPuntos[i+1][0].iD > id){     //aqui el +1 me
hace salir de la tabla con su consecuente error.
                    for (int k=0; k<columnas; k++){
                        if (mapPuntos[i][k].iD == id){
                            posicion[0]=i;
                            posicion[1]=k;
                            encontrado = true;
                            break;
                        }
                    }
```

```
                    break;
                }
            }

        }

        if (encontrado == false){
            int i=filas-1;
            for (int k=0; k<columnas; k++){
                if (mapPuntos[i][k].iD == id){
                    posicion[0]=i;
                    posicion[1]=k;
                }
            }
        }

    return posicion;
    }

    //devuelve el punto cuya id se pide
    public static Punto buscaPuntoId(int id){
        int[] posicion = new int[2];

        posicion = buscaID(id);

        return mapPuntos[posicion[0]][posicion[1]];
    }


    //rellena el campo "vecinos" del objeto Punto
    public static void setVecinos(Punto punto){
        int aux = 0;
        int i = punto.posX;
        int k = punto.posY;

        if ((i - 1) >= 0 && mapPuntos[i-1][k].pasable == 1){
            punto.vecinosPasables[aux] = mapPuntos[i-1][k].iD;
            aux++;
        }

        if ((k + 1) < columnas && mapPuntos[i][k+1].pasable == 1){
            punto.vecinosPasables[aux] = mapPuntos[i][k+1].iD;
            aux++;
        }

        if ((i + 1) < filas && mapPuntos[i+1][k].pasable == 1){
            punto.vecinosPasables[aux] = mapPuntos[i+1][k].iD;
            aux++;
        }

        if ((k - 1) >= 0 && mapPuntos[i][k-1].pasable == 1){
            punto.vecinosPasables[aux] = mapPuntos[i][k-1].iD;
        }

    }

    public static void setAllVecinos(){
        for (int i=0; i<filas;i++){
            for (int k=0; k<columnas; k++){
                setVecinos(mapPuntos[i][k]);
            }
```

```
        }
    }


    // Calcula el valor de H para un punto
    public static int valorH(Punto punto, Punto destino){

        int distX = Math.abs(punto.posX - destino.posX);
        int distY = Math.abs(punto.posY - destino.posY);

        return distX+distY;
    }

    public static int[] getVecinos(int id){

        Punto punto = buscaPuntoId(id);
        int aux = 0;

        for (int i = 0; i<punto.vecinosPasables.length; i++){
            if (punto.vecinosPasables[i]!=0) aux++;
            else break;
        }

        int [] vecinos = new int[aux];

        for (int i=0; i<aux; i++){
            vecinos[i]=punto.vecinosPasables[i];
        }

        return vecinos;

    }

    //devuelve la id de una posici□ada en un vector.
    public static int queId(int[] posicion){
        int id = mapPuntos[posicion[0]][posicion[1]].iD;
        return id;


    }

}
```

3.3.3 Localización

En esta clase se almacenan los valores correspondientes a las RSSI medidas en los distintos puntos a recorrer de nuestro mapa, así como los métodos encargados de comparar el vector de potencias recibido con los de referencia y tomar la decisión de en qué posición se encuentra el usuario.

Tomando como referencia la posición actual en la que se cree que se encuentra el usuario dicho método selecciona tan sólo los vectores de potencias correspondientes a su posición y a la de los vecinos ahorrando así cálculos innecesarios.

```
package com.example.bshoppingassistant;
import java.util.ArrayList;
```

```
public class Localizacion {



    static int[] arrayId = {12, 17, 21, 22, 23, 24, 25, 26, 28};
     //static List listId = Arrays.asList(arrayId);
    static ArrayList<Integer> listId = new ArrayList<Integer>();
    static{
        for (int i=0; i<arrayId.length;i++) listId.add(arrayId[i]);
    }


    static int[][] rssiMedia = {
            {-90, -58, -71},
            {-83, -74, -63},
            {-68, -84, -61},
            {-80, -72, -63},
            {-85, -83, -56},
            {-86, -85, -57},
            {-74, -93, -78},
            {-61, -87, -77},
            {-85, -86, -55}
    };

    public static int difCuadrado(int[] vector1, int[] vector2){
        int diferencia = 0;

        if (vector1.length==vector2.length){
            for (int i=0; i<vector1.length; i++){
                int aux = (int) Math.pow(Math.abs(vector1[i]-
vector2[i]),2);
                diferencia += aux;
            }
        }

        return diferencia;

    }




    public static int localiza(int posicion, int[] vectorPrueba){


        int[] vecinos = Buscamino.getVecinos(posicion);
        int l = vecinos.length + 1;

        int[] todos = new int[l];

        for (int i=0; i<vecinos.length; i++){
            todos[i]=vecinos[i];
        }
        todos[l-1]=posicion;

        int[] arrayDif = new int[l];

        int [][] rssiVecinos = new int[l][];
```

```
        //rellenamos la tabla incluyendo el punto actual


        rssiVecinos[l-1] = rssiMedia[listId.indexOf(posicion)];
        for (int i=0; i<l; i++)
            rssiVecinos[i] = rssiMedia[listId.indexOf(todos[i])];

        for (int i=0; i<l; i++){
            arrayDif[i] = difCuadrado(vectorPrueba, rssiVecinos[i]);
        }

        int puntoMenor = buscaIndMenor (arrayDif);
        return todos[puntoMenor];
    }



    public static int buscaIndMenor (int[] array){
        int l = array.length;
        int menor = array[0];
        int puntero = 0;
        if (l!=1){
            for (int i=1; i<l; i++){
                if (array[i]<menor){
                    menor = array[i];
                    puntero = i;
                }
            }
        }

        return puntero;
    }

}
```

3.3.5 Ordenador

En esta clase se almacenan los métodos encargados de ordenar los elementos presentes en la lista de la compra de una manera eficiente para con ello después crear la ruta a seguir. Puede ordenarlos tomando como referencia la entrada o un punto dado.

Cuando se utiliza un punto dado este será la posición actual al recalcular la ruta durante el guiado.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;

public class Ordenador {

    private static ArrayList<String> open = new ArrayList<String>();
    private static ArrayList<String> closed = new ArrayList<String>();


    // m鴫do para ordenar por defecto

    public static String ordena(String compra){
```

```
        String ordenado;
        String ultimo = "Entrada";
        int aux;
        int puntero;
        int temp;

        open = apoyo.stringToAlist(compra);
        closed.clear();

        if (open.contains("Entrada")){
            int borrar =open.indexOf("Entrada");
            open.remove(borrar);}

        if (open.contains("Salida")){
            int borrar =open.indexOf("Salida");
            open.remove(borrar);
        }

        closed.add(ultimo);

        if (!open.isEmpty()){
            do{
                aux = distancia(ultimo, open.get(0));
                puntero = 0;
                for (int i=1; i<open.size();i++){
                    temp = distancia(ultimo, open.get(i));

                    if (temp < aux){
                        aux = temp;
                        puntero = i;
                    }

                }

                ultimo = (open.get(puntero));
                ponClosed(puntero);


            } while (!open.isEmpty());
        }

        closed.add("Salida");


        ordenado = apoyo.alistToString(closed);


        return ordenado;
    }

    //m鴨do para ordenar desde un punto y no la entrada

    public static String ordena(String compra, String puntoAuxiliar){
        String ordenado;
        String ultimo = puntoAuxiliar;
        int aux;
        int puntero;
        int temp;

        open = apoyo.stringToAlist(compra);
        closed.clear();
```

```
            if (open.contains(puntoAuxiliar)){
                int borrar =open.indexOf(puntoAuxiliar);
                open.remove(borrar);}

            if (open.contains("Entrada")){
                int borrar =open.indexOf("Entrada");
                open.remove(borrar);}

            if (open.contains("Salida")){
                int borrar =open.indexOf("Salida");
                open.remove(borrar);
            }

            closed.add(ultimo);

            if (!open.isEmpty()){
                do{
                    aux = distancia(ultimo, open.get(0));
                    puntero = 0;
                    for (int i=1; i<open.size();i++){
                        temp = distancia(ultimo, open.get(i));

                        if (temp < aux){
                            aux = temp;
                            puntero = i;
                        }

                    }

                    ultimo = (open.get(puntero));
                    ponClosed(puntero);


                } while (!open.isEmpty());
            }

            closed.add("Salida");


            ordenado = apoyo.alistToString(closed);


            return ordenado;
        }

        private static void ponClosed(int posicion){
            closed.add(open.get(posicion));
            open.remove(posicion);
        }

        public static int distancia(String inicio, String destino){
            int ix = Posiciones.lugar.get(inicio)[0];
            int iy = Posiciones.lugar.get(inicio)[1];
            int dx = Posiciones.lugar.get(destino)[0];
            int dy = Posiciones.lugar.get(destino)[1];


            return (Buscamino.buscaCamino(ix, iy, dx, dy).size()-1);
        }
    }
```

3.3.6 Posiciones

En esta clase se almacena tanto el mapa base de la superficie como las posiciones de los distintos elementos a localizar. De paso se inicializa la posición actual asumiendo que se encuentra el usuario en la entrada del recinto.

```
package com.example.bshoppingassistant;

import java.util.HashMap;

public class Posiciones {

    public static int[][] mapa = {
        {0,0,1,0,0},
        {0,0,1,0,0},
        {0,1,1,1,1},
        {1,1,0,1,0},

    };

    public static void setActual(int[] posicion){
        lugar.put("actual", posicion);
    }

    public static final HashMap<String, int[]> lugar;
    static
    {

        lugar = new HashMap<String, int[]>();

        int[] actual = {2, 1}; //inicializado en la entrada
        lugar.put("actual", actual);

        int[] entrada = {2,1};
        lugar.put("Entrada",entrada);

        int[] salida = {2,1};
        lugar.put("Salida",salida);

        int[] pollo = {3,0};
        lugar.put("Pollo",pollo);

        int[] ternera = {3,1};
        lugar.put("Ternera",ternera);

        int[] zanahoria = {2,4};
        lugar.put("Zanahoria",zanahoria);

        int[] puerro = {2,3};
        lugar.put("Puerro",puerro);

        int[] leche = {3,1};
        lugar.put("Leche",leche);

        int[] yogurt = {3,1};
        lugar.put("Yogurt",yogurt);

        int[] manzana = {0,2};
        lugar.put("Manzana",manzana);
```

```
        int[] platano = {1,2};
        lugar.put("Platano",platano);

        int[] salmon = {2,2};
        lugar.put("Salmon",salmon);

        int[] sardinas = {2,2};
        lugar.put("Sardinas",sardinas);

    }

}
```

3.3.7 Punto

La clase Punto se utilizó para definir las posiciones en la clase Buscamino y otras. Posee varios campos entre los que almacena su ID, si se puede caminar por él, su posición y la ID de los vecinos que puedan ser accesibles desde él. Para inicializarlo y distinguirlo de los puntos que se van a tratar en caso de que no sea "pasable" algunos de sus valores se inicializan con "-1".

```
package com.example.bshoppingassistant;

public class Punto {

    public int iD;
    public int pasable;
    public int posX;
    public int posY;
    public int[] vecinosPasables = new int[4];

    //para calculo del camino
    int g;
    int h;
    int f;


    public Punto(){

        this(0,0,0,0);

    }


    public Punto(int iD, int pasable, int posX, int posY){

        this.iD = iD;
        this.pasable= pasable;
        this.posX=posX;
        this.posY=posY;
        this.g=-1;
        this.h=-1;
        this.f=-1;

    }

}
```

3.3.8 Ruta

Esta clase posee los métodos necesarios para, mediante el uso de Buscamino, crear la ruta punto a punto que ha de recorrer el usuario dada una lista de productos.

```
package com.example.bshoppingassistant;

import java.util.ArrayList;

public class Ruta {

    static int ix;
    static int iy;
    static int dx;
    static int dy;


    public static ArrayList<Integer> creaRuta(String lista){
        int aux;


        //ArrayList<Punto> temp = new ArrayList<Punto>();
        ArrayList<Punto> auxiliar = new ArrayList<Punto>();
        ArrayList<Integer> idRuta = new ArrayList<Integer>();
        ArrayList<String> aList = apoyo.stringToAlist(lista);

        apunta(aList.get(0),aList.get(1));

        //temp = Buscamino.buscaCamino(ix, iy, dx, dy);

        for (int i=0; i<aList.size()-1;i++){

            apunta(aList.get(i), aList.get(i+1));
            auxiliar = Buscamino.buscaCamino(ix, iy, dx, dy);

            for (int k = 0; k < auxiliar.size(); k++){
                aux = auxiliar.get(k).iD;
                idRuta.add(aux);
            }
        }

        idRuta = noRep(idRuta);

        return idRuta;



    }

    private static void apunta(String inicio, String destino){

        ix = Posiciones.lugar.get(inicio)[0];
        iy = Posiciones.lugar.get(inicio)[1];
        dx = Posiciones.lugar.get(destino)[0];
        dy = Posiciones.lugar.get(destino)[1];

    }
```

```
    private static ArrayList<Integer> noRep(ArrayList<Integer> lista){

        for (int i=0; i<lista.size()-1 ; i++){
            if (lista.get(i)==lista.get(i+1)){
                do{
                    lista.remove(i+1);

                }while(lista.get(i)==lista.get(i+1));
            }
        }

        return lista;

    }

}
```

3.3.9 Tablas

En este método tratamos la información recogida por los sensores y creamos el vector de valores RSSI con el que trabajará "Localización".

```
package com.example.bshoppingassistant;


import java.util.ArrayList;
import java.util.HashMap;

public class Tablas {

    public static String baliza1 = "D0:39:72:A0:8D:FD";
    public static String baliza2 = "D0:39:72:A0:88:C8";
    //public static String localWifi = "00:13:10:CD:C4:82";
    public static String localWifi = "WLan";

    public static HashMap<String, ArrayList<Integer>> valores = new
HashMap<String, ArrayList<Integer>>();


    public static void iniciar(){
        valores.put(baliza1, new ArrayList<Integer>());
        valores.put(baliza2, new ArrayList<Integer>());
        valores.put(localWifi, new ArrayList<Integer>());
    }

    public static void limpiar(){
        valores.get(baliza1).clear();
        valores.get(baliza2).clear();
        valores.get(localWifi).clear();
    }

    public static int media(String dispositivo){
        ArrayList<Integer> aux = valores.get(dispositivo);
        int sum = 0;
        int sinValor = -99;
```

```
            if (!aux.isEmpty()){
                if (aux.size()==0) return sinValor;

                for (int i=0; i<aux.size(); i++){
                    sum += aux.get(i);
                }

                return sum/aux.size();
            }
            else return sinValor;
        }

        public static void anade(String dispositivo, int valor){
            ArrayList<Integer> aux = valores.get(dispositivo);
            aux.add(valor);
        }

        public static int[] getVectorMedia(){
            int[] salida =
    {media(baliza1),media(baliza2),media(localWifi)};
            return salida;
        }

    }
```

3.4 Layouts

Los Layouts son la parte encargada de la representación gráfica de las actividades. Donde se colocarán los botones y se definirán sus tamaños y el de sus textos, etc.

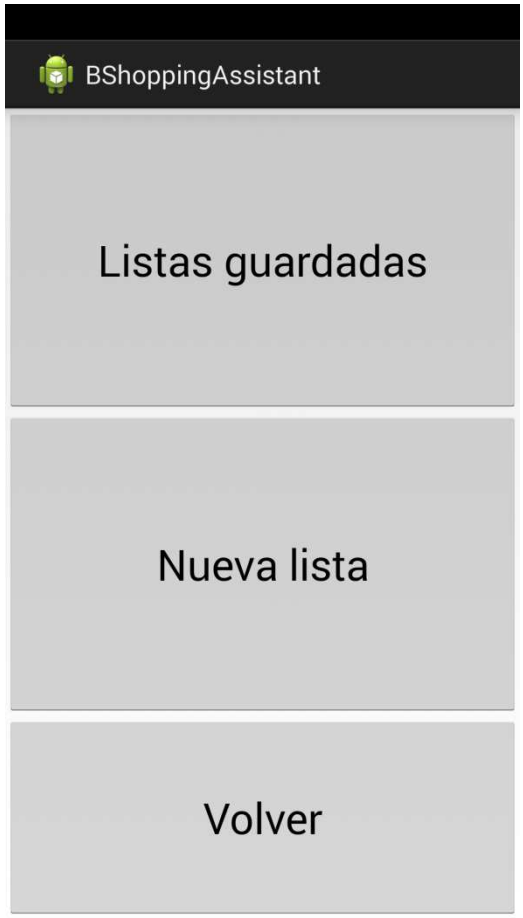
3.4.1 MainActivity


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
```

```
            android:layout_height="0dp"
            android:layout_weight="3"
            android:text="@string/boton1.1"
            android:textSize="30sp"
             />

        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:onClick="muestraRuta"
            android:layout_weight= "3"
            android:text="@string/boton1.2"
            android:textSize="30sp"
             />

    </LinearLayout>
```

3.4.2 MenuLista

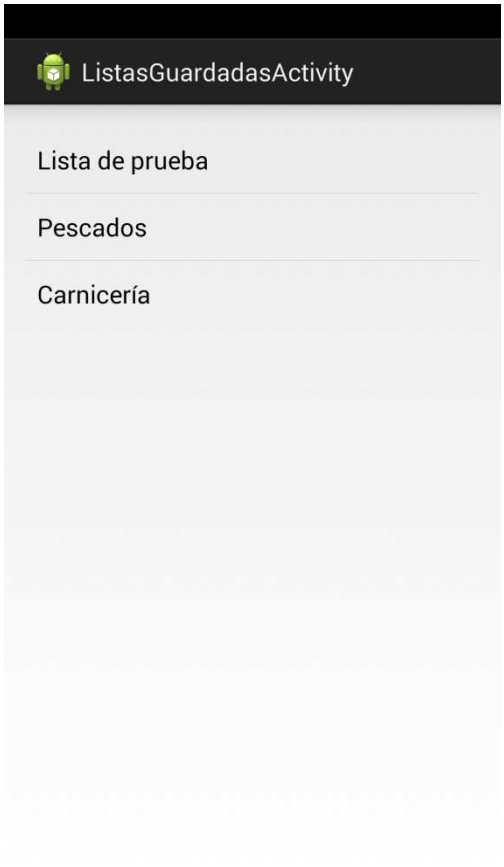
Es la asociada a la actividad ActivityScr02




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
```

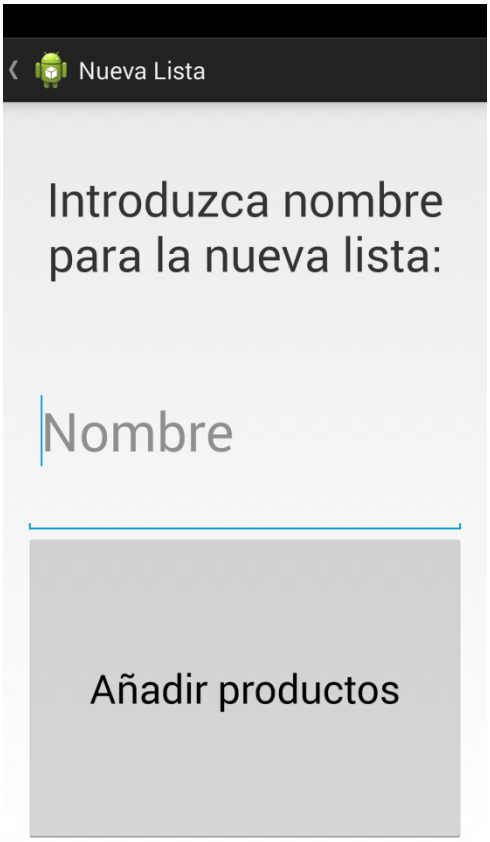
```
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="listasGuardadas"
        android:text="@string/scr02.boton1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="crearLista"
        android:text="@string/scr02.boton2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="30sp"
        android:onClick="cerrar"
        android:text="@string/volver" />

</LinearLayout>
```

3.4.3 ListasGuardadasActivity




```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
     >


    <ListView
                        android:id="@+id/mListViewVerListas"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content" >
                    </ListView>


</RelativeLayout>
```

3.4.4 NuevaLista




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".NuevaLista" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:gravity="center_vertical|center_horizontal"
        android:text="@string/nuevaLista.intNombre"
        android:textSize="35sp" />

    <EditText
        android:id="@+id/nuevoNombre"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="40sp"
        android:hint="@string/nombre" >
```

```
        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/botonAnadir"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="abreLista"
        android:text="@string/anadirProductos"
         />

    </LinearLayout>
```

3.4.5 ListaBase

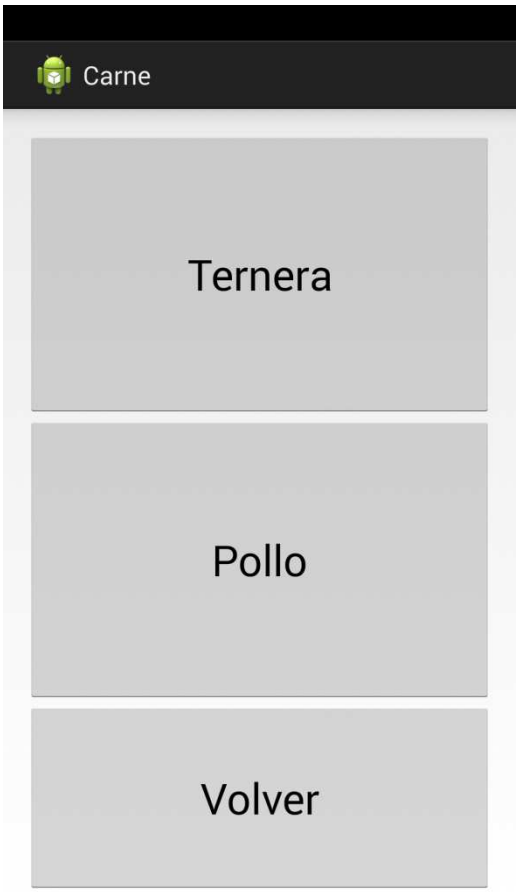

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".ListaBase" >
```

```
<Button
    android:id="@+id/button1"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="abreCarne"
    android:text="@string/carne" />

<Button
    android:id="@+id/button2"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="abrePescado"
    android:text="@string/pescado" />

<Button
    android:id="@+id/button3"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="abreLacteos"
    android:text="@string/lacteos" />

<Button
    android:id="@+id/button4"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="abreVerduras"
    android:text="@string/verduras" />

<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="abreFrutas"
    android:text="@string/frutas" />

<Button
    android:id="@+id/button7"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:textSize="30sp"
    android:onClick="verLista"
    android:text="@string/verLista" />

<Button
    android:id="@+id/button8"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
```

```
        android:textSize="30sp"
        android:onClick="borrarLista"
        android:text="@string/borrarLista" />

    <Button
        android:id="@+id/button6"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="20sp"
        android:onClick="cerrar"
        android:text="@string/volver" />

</LinearLayout>
```

3.4.6 Carne

En los layouts respectivos de las siguientes categorías de alimentos se omitirá la imagen preliminar por ser en esencia la misma pero con otros productos escritos.




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Carne" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadeTernera"
        android:text="@string/ternera" />

    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadePollo"
        android:text="@string/pollo" />

    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="30sp"
        android:onClick="cerrar"
        android:text="@string/volver" />

</LinearLayout>
```

3.4.7 Pescado

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Pescado" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
```

```
            android:onClick="anadeSalmon"
            android:text="@string/salmon" />

        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="3"
            android:textSize="30sp"
            android:onClick="anadeSardinas"
            android:text="@string/sardinas" />

        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="2"
            android:textSize="30sp"
            android:onClick="cerrar"
            android:text="@string/volver" />

    </LinearLayout>
```

3.4.8 Lacteos

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Lacteos" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadeLeche"
        android:text="@string/leche" />

    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadeYogurt"
        android:text="@string/yogurt" />

    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
```

```
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="30sp"
        android:onClick="cerrar"
        android:text="@string/volver" />

</LinearLayout>
```

3.4.9 Verduras

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Verduras" >

    <Button
        android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadeZanahoria"
        android:text="@string/zanahoria" />

    <Button
        android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:textSize="30sp"
        android:onClick="anadePuerro"
        android:text="@string/puerro" />

    <Button
        android:id="@+id/button3"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:textSize="30sp"
        android:onClick="cerrar"
        android:text="@string/volver" />

</LinearLayout>
```

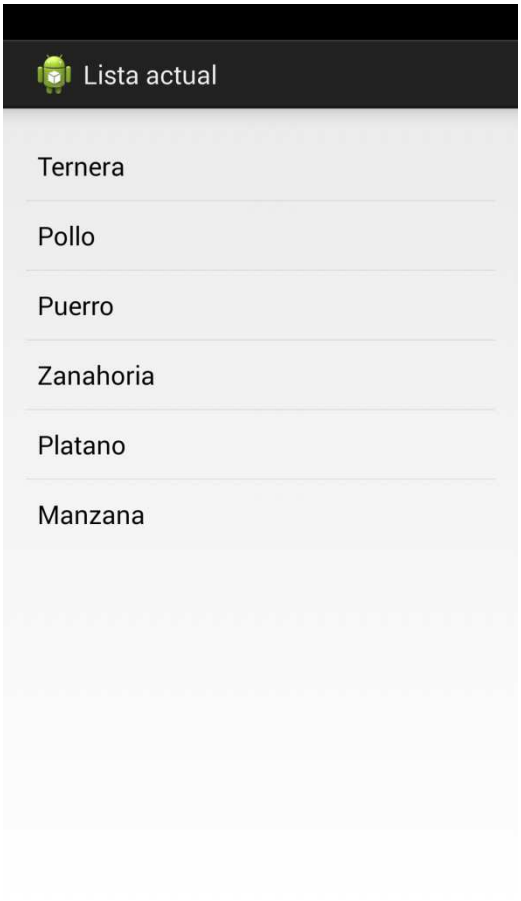
3.4.10 Frutas

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context=".Frutas" >

        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="3"
            android:textSize="30sp"
            android:onClick="anadeManzana"
            android:text="@string/manzana" />

        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="3"
            android:textSize="30sp"
            android:onClick="anadePlatano"
            android:text="@string/platano" />

        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="2"
            android:textSize="30sp"
            android:onClick="cerrar"
            android:text="@string/volver" />

    </LinearLayout>
```

3.4.11 VerLista

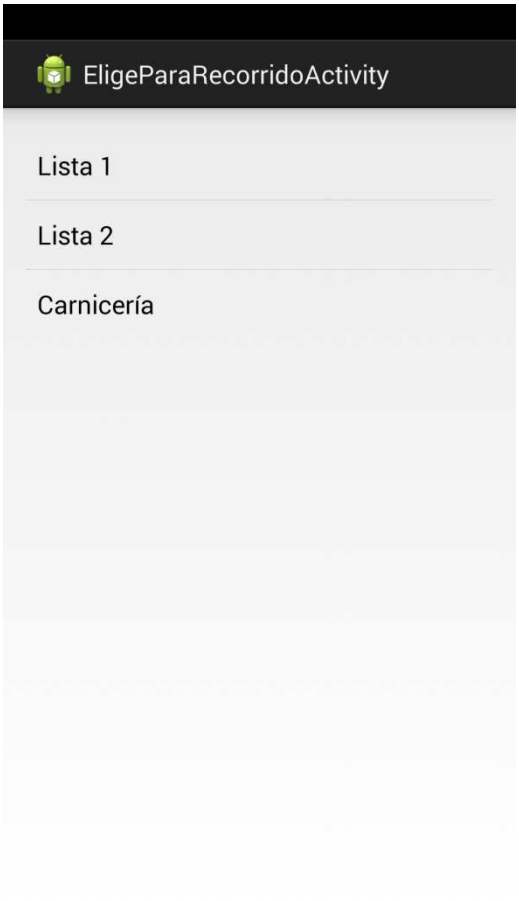
En esta pantalla se mostrarán los elementos que se encuentren añadidos a la lista actual.




```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".VerLista" >


    <ListView
                        android:id="@+id/mListViewVerLista"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content" >
                    </ListView>



</RelativeLayout>
```

3.4.12 EligeParaRecorrido

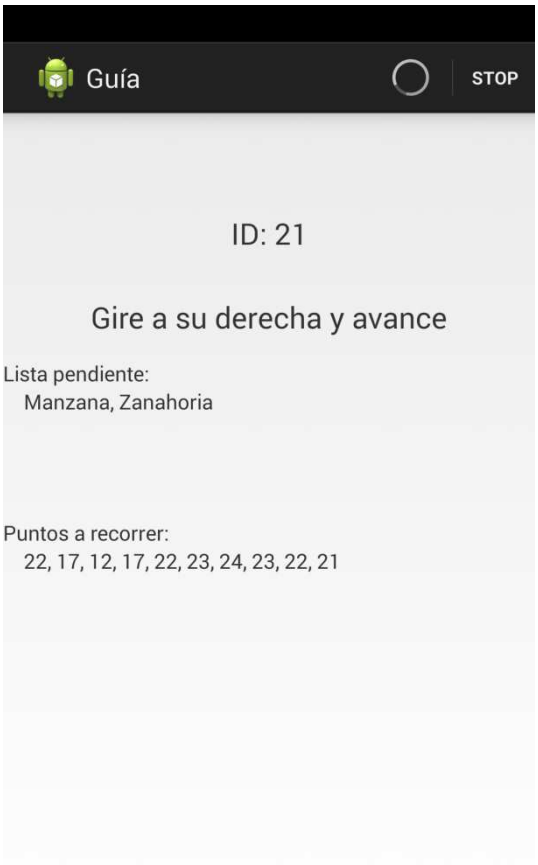
En esta pantalla se mostrarán las posibles listas para iniciar un recorrido con ellas.




```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
     >


    <ListView
                        android:id="@+id/mListViewVerListas"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content" >
                    </ListView>



</RelativeLayout>
```

3.4.13 GuiaActivity

Esta pantalla será la que muestre la siguiente indicación a seguir. De paso muestra la ID de la posición actual, los elementos de la lista que restan por recoger y los puntos que se deben recorrer aún.




```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/posicionActual"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="68dp"
        android:text="ID Posicion"
        android:textSize="20sp" />

    <TextView
        android:id="@+id/indicaciones"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/posicionActual"
        android:layout_centerHorizontal="true"
```

```
        android:layout_marginTop="30dp"
        android:text="Indicación"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:text="Puntos a recorrer:" />

    <TextView
        android:id="@+id/recorrer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="14dp"
        android:text="Cargando..." />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/indicaciones"
        android:layout_marginTop="15dp"
        android:text="Lista pendiente:" />

    <TextView
        android:id="@+id/cosaslista"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/recorrer"
        android:layout_below="@+id/textView2"
        android:text="Lista" />

    </RelativeLayout>
```

3.5 Strings.xml

Aquí se almacenan Strings a los que hacer referencia.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">BShoppingAssistant</string>
    <string name="guia">Guía</string>
    <string name="action_settings">Settings</string>
    <string name="volver">Volver</string>
    <string name="verLista">Ver lista</string>
    <string name="borrarLista">Borrar lista</string>
    <string name="boton1.1">Lista de la compra</string>
    <string name="boton1.2">Comenzar compra</string>
    <string name="boton1.3">Salir</string>
    <string name="boton1.4" />
```

```
    <string name="borraListas">Borrar listas</string>
    <string name="scr02.boton1">Listas guardadas</string>
    <string name="scr02.boton2">Nueva lista</string>
    <string name="scr02.boton3">Seleccionar favoritos</string>
    <string name="nuevaLista.intNombre">Introduzca nombre para la nueva
lista:</string>
    <string name="anadirProductos">Añadir productos</string>
    <string name="nombre">Nombre</string>
    <string name="carne">Carne</string>
    <string name="ternera">Ternera</string>
    <string name="pollo">Pollo</string>
    <string name="pescado">Pescado</string>
    <string name="salmon">Salmón</string>
    <string name="sardinas">Sardinas</string>
    <string name="lacteos">Lácteos</string>
    <string name="leche">Leche</string>
    <string name="yogurt">Yogurt</string>
    <string name="verduras">Verduras</string>
    <string name="zanahoria">Zanahoria</string>
    <string name="puerro">Puerro</string>
    <string name="frutas">Frutas</string>
    <string name="manzana">Manzana</string>
    <string name="platano">Plátano</string>
    <string name="title_activity_nueva_lista">Nueva Lista</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_lista_base">Elementos</string>
    <string name="title_activity_carne">Carne</string>
    <string name="title_activity_pescado">Pescado</string>
    <string name="title_activity_lacteos">Lacteos</string>
    <string name="title_activity_verduras">Verduras</string>
    <string name="title_activity_frutas">Frutas</string>
    <string name="title_activity_ver_lista">Lista actual</string>
    <string name="title_activity_ruta">Ruta</string>
    <string name="title_activity_muestra_ruta">MuestraRuta</string>
    <string
name="title_activity_listas_guardadas">ListasGuardadasActivity</string>
    <string
name="title_activity_elige_para_recorrido">EligeParaRecorridoActivity</string
>
    <string name="title_activity_guia">GuiaActivity</string>
    <string name="ble_not_supported">BLE is not supported</string>
    <string name="error_bluetooth_not_supported">Bluetooth not
supported.</string>
    <string name="unknown_device">Unknown device</string>


    <!-- Menu items -->
    <string name="menu_connect">Connect</string>
    <string name="menu_disconnect">Disconnect</string>
    <string name="menu_scan">Scan</string>
    <string name="menu_stop">Stop</string>

      </resources>
```

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

Adaptación de un entorno inteligente mediante balizas BLE para invidentes

PLIEGO DE CONDICIONES

Autor: Guillermo García de la Fuente
Tutor: Ignacio R. Matías Maestro
Pamplona, julio 2014

Contenido

1. Introducción

En este documento se procederá a describir las condiciones necesarias para el manejo y correcto funcionamiento de la aplicación en una superficie.

2. Condiciones técnicas

2.1 Dispositivo móvil

Será necesario un dispositivo con sistema operativo Android 4.3 (API level 18) o superior y capacidad para recibir señales tanto Bluetooth como del protocolo 802.11.

Actualmente existe una amplia gama que cumple los requisitos y el producto ha sido testado sobre un Samsung Galaxy S4 sin problemas

2.2 Balizas

Las balizas deberán emitir una señal Bluetooth Low Energy con la que pueda ser identificada por su dirección MAC por un dispositivo capaz de recibirla.

Se espera de ellas un alcance de hasta 50 metros en espacio abierto, asumiendo que el límite de alcance se ve marcado por una recepción de -100 dBm.

2.3 Router inalámbrico

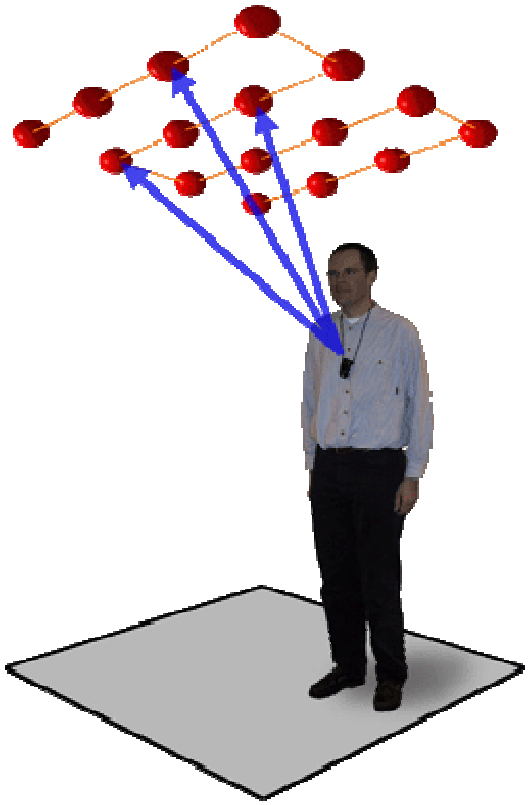
El router inalámbrico no requiere de ninguna funcionalidad especial más que la de crear una red de área local inalámbrica de la que poder deducir su potencia en determinados puntos del área a cubrir.

3. Montaje y ejecución

Para poder preparar un Entorno Inteligente Adaptado como el sugerido será necesario estudiar el área a tratar.

En primer lugar el área será dividida en cuadrículas que determinarán las posibles posiciones del usuario y se les asignará una determinada ID. Tras ello se colocará un número de balizas suficientes distribuidas por la superficie de manera que el usuario tenga siempre la visibilidad de al menos 3 de ellas.

La posición de las balizas será muy determinante dado lo susceptible que es este tipo de señal a las atenuaciones por obstáculos. Es por ello que lo ideal sería colocarlas en el techo o en puntos altos que puedan cubrir una buena área y no ser su señal obstaculizada por otras personas caminando cerca del usuario. Un ejemplo sería cómo se distribuían los sensores del sistema BAT.



Una vez colocadas las balizas debe procederse a las lecturas y mediciones que nos llevarán a crear los vectores de potencias para cada punto. Para ello sea mediante una aplicación específica o algún otro medio se tomará la media de la señal recibida en el punto durante al menos 1 minuto y medida en distintas orientaciones para tratar de compensar la posible atenuación del cuerpo del usuario.

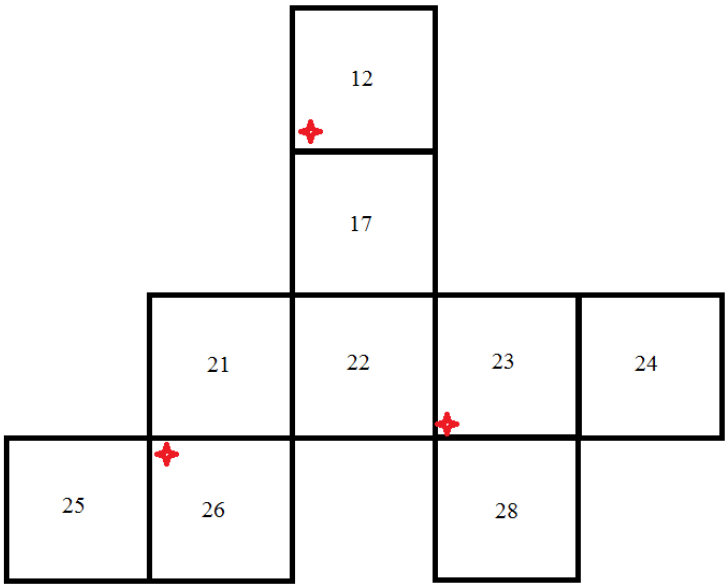
Cuando se cuente con todas las medidas, éstas se añadirán mediante dos Arrays a la clase "Localizacion" de la aplicación. En el primero se colocarán las respectivas ID de los puntos analizados. El segundo Array será bidimensional y almacenará los

vectores de potencias de los respectivos puntos en el mismo orden que fueron introducidos en el primero.

Hecho esto se introducirán los datos de la posición de los puntos de interés (productos, entrada, salida e inicialización de la posición del usuario) en la clase "Posiciones", así como la tabla respectiva de unos y ceros que representará el mapa del lugar siendo los "0" espacio no accesible y los "1" espacio accesible.

Ya sólo resta instalar la aplicación en un dispositivo compatible e iniciarla.

A continuación se muestra un ejemplo de la colocación de 3 balizas para cubrir un área de 25 metros cuadrados:

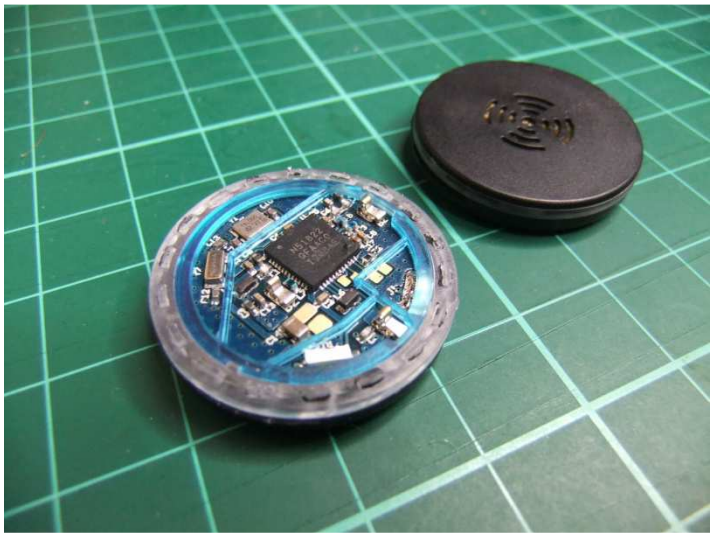


4. Pruebas del material

El material debe cumplir no solo con las condiciones técnicas sino que a su llegada se deberá comprobar lo siguiente:

4.1 Balizas

Deberán emitir correctamente cumpliendo las condiciones de recepción establecidas y con una dirección única para cada una de ellas. De este modo se evitarán problemas añadiendo su respectiva potencia al vector.



4.2 Dispositivo móvil

Debe comprobarse que el dispositivo móvil puede ejecutar la aplicación sin que ésta finalice en error. Así mismo deberá comprobarse que es capaz de recibir los dos tipos de señales especificados (Bluetooth y 802.11) e interactuar con los dispositivos BTLE recibiendo sus señales y siendo capaz de indicar la potencia.

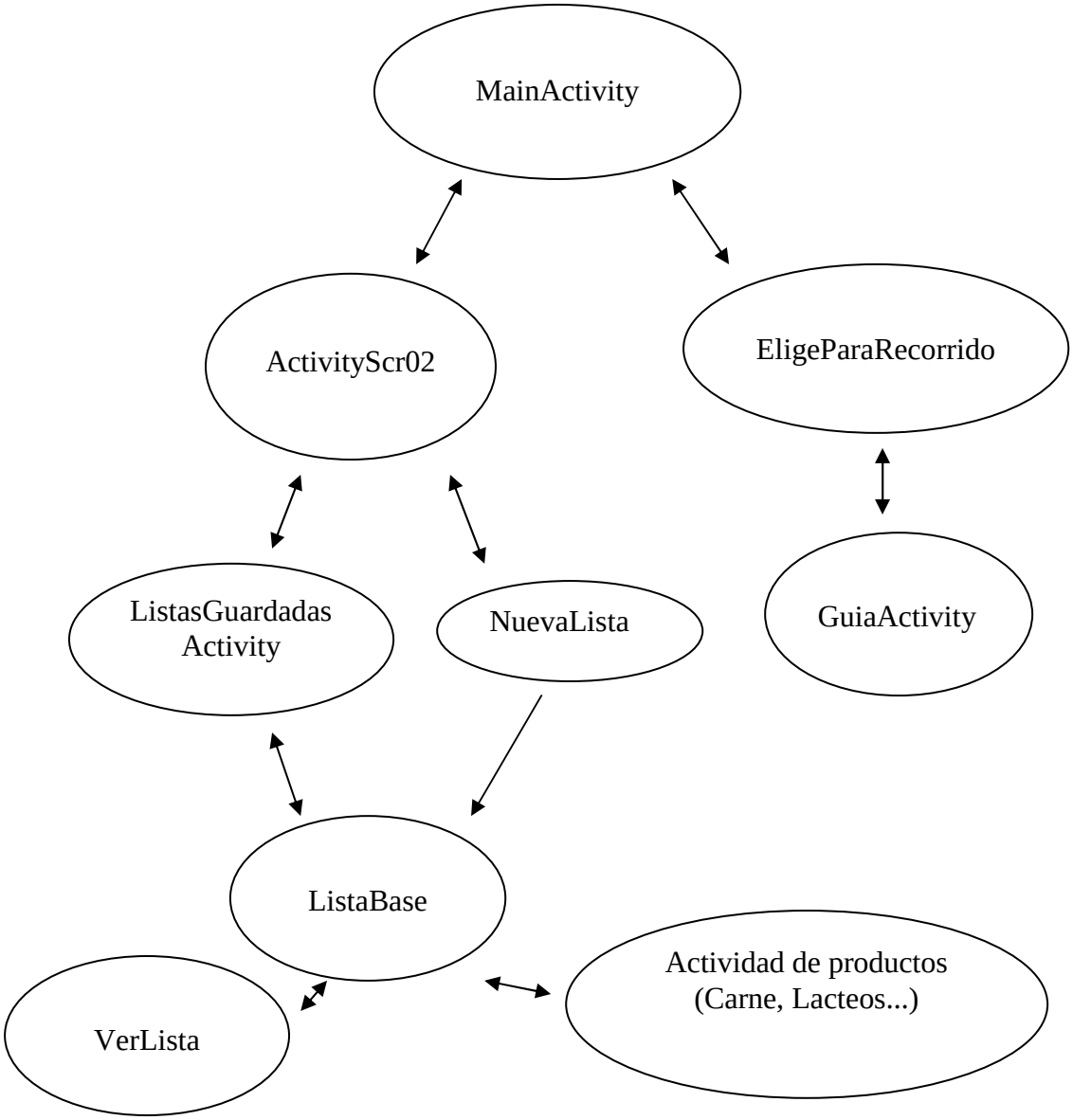
5. Uso de la aplicación

Se dividirá este apartado en dos secciones. El uso de la aplicación y el modo de actualizarla en caso de mover un producto o modificar el mapa.

5.1 Manejo

En primer lugar, la aplicación ha sido creada para cubrir la mayor parte de la pantalla con botones. Estos botones tienen una función asociada al evento "LongClick" que hace que al ser pulsados de manera prolongada lean al usuario el texto asociado al botón. De este modo para identificar el botón (o elemento de la lista) con el que se quiera interactuar tan sólo se ha de ir pulsando distintas zonas de la pantalla hasta localizar el objeto deseado.

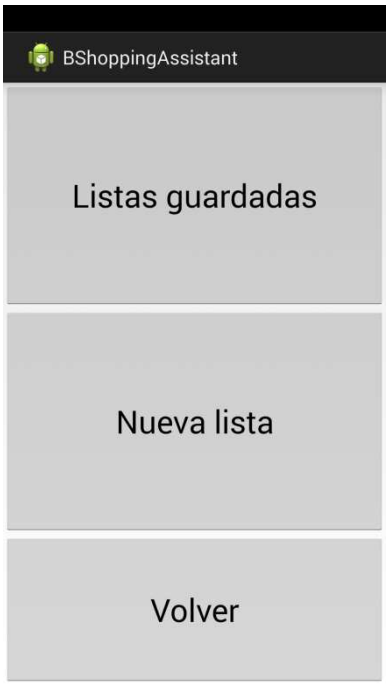
El diagrama de flujo entre las actividades es el siguiente, indicando entre cuales se puede navegar pasando de una a otra:



Comenzando en MainActivity nos encontramos con la siguiente pantalla:

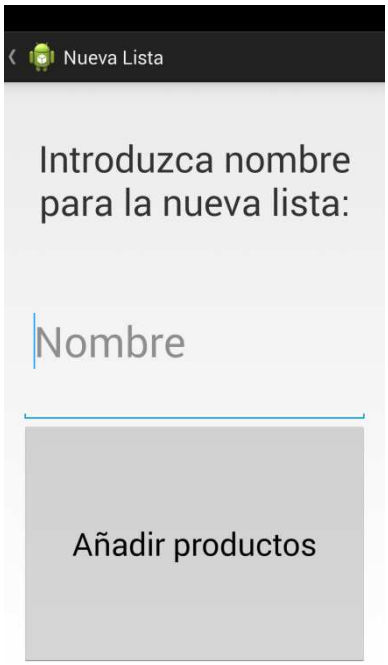
En ella los dos botones nos muestran las dos posibles actividades a las que podemos acceder. Como aún no hay ninguna lista guardada si nos dirigimos a la actividad "EligeParaRecorrido" ésta nos mostrará una única entrada diciendo que no hay ninguna lista y que no nos permitirá hacer nada más que volver a la actividad anterior.

Así pues haremos un clic simple en "Lista de la compra" para disponernos a crear una. Llegaremos a la siguiente pantalla:



Desde esta actividad podemos acceder a las listas que hayamos creado anteriormente o crear una nueva. Si pulsamos sobre "Listas guardadas" se nos leerán las listas existentes y se mostrarán en pantalla. Como se dijo anteriormente con una pulsación prolongada se leerá la lista y con una simple se accederá a ella.

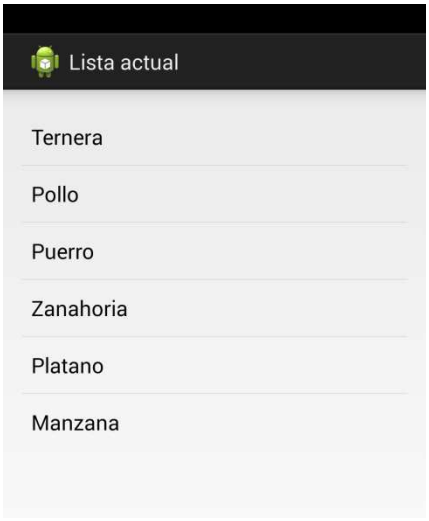
Como aún no hay ninguna seleccionaremos "Nueva lista".



Introduciremos el nombre deseado y haremos clic en "Añadir productos". Los teclados Android actuales incluyen la funcionalidad de introducción de texto mediante voz, por lo que no se ha implementado dicha función.

Tras esto llegaremos a una nueva actividad denominada "ListaBase". En ella se podrá comprobar, modificar o eliminar la lista creada o elegida. Para añadir productos a la lista haremos clic en una de las categorías de productos (Carne, lácteos...), lo que nos llevará a una nueva Actividad con los productos disponibles para dicha categoría. Elegiremos el deseado y volveremos.

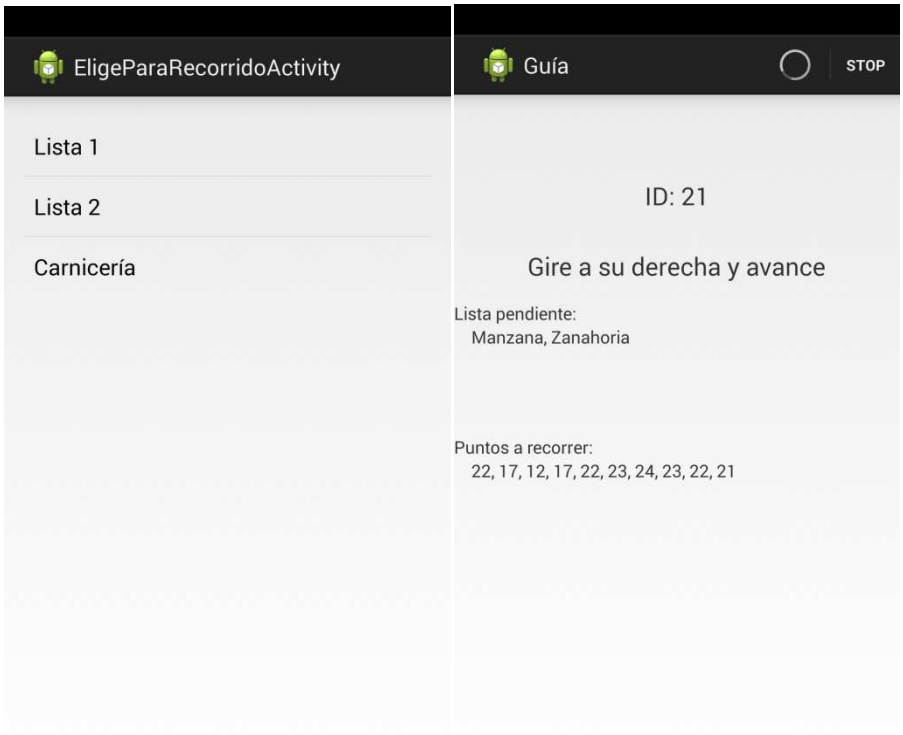
Si queremos comprobar los elementos de la lista actual seleccionaremos "Ver lista", lo que hará que se abra una nueva actividad con la lista de productos actual. Dicha actividad nos leerá los productos ya ordenados con una ruta optimizada desde la entrada del establecimiento.



Como en todas las listas puede leerse el elemento deseado dejando pulsada la pantalla donde éste se encuentra, pero a diferencia de las demás en este caso si se realiza un clic simple sobre él será eliminado y la lista se leerá de nuevo al usuario.

Una vez que la lista haya sido debidamente rellenada podemos volver a la actividad principal y elegir "Comenzar compra".

Se nos mostrarán las listas que hayamos creado y cuando elijamos una se iniciará la actividad "GuiaActivity". En dicha actividad comenzaremos a recibir indicaciones para ir a por el siguiente producto en nuestra lista. Las seguiremos y una vez que lleguemos a ese punto la aplicación nos indicará "Recoja ***", siendo "***" los productos de nuestra lista que se encuentren en tal posición.



La actividad finalmente nos llevará a la salida del establecimiento cuando ya hayamos recorrido todos los puntos de nuestros productos y finalizará automáticamente.

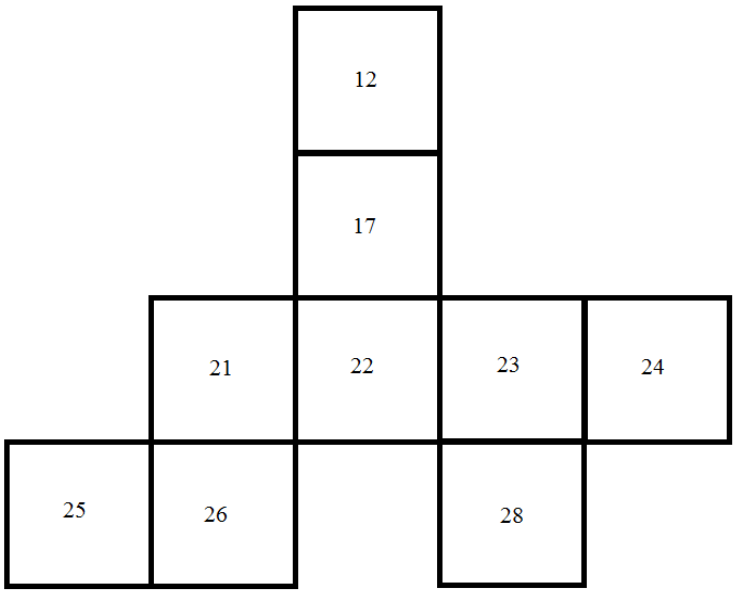
5.2 Introducción del mapa y posiciones

Para introducir el mapa de unos y ceros correspondiente a la superficie a tratar deberemos hacerlo en la clase "Posiciones" de nuestra aplicación.

El mapa se introducirá con el siguiente formato:

```
public static int[][] mapa = {
            {0,0,1,0,0},
            {0,0,1,0,0},
            {0,1,1,1,1},
            {1,1,0,1,0},

            };
```

Dicho ejemplo representa el siguiente mapa:



Y las posiciones respectivas se colocarán teniendo en cuenta que los pares de números que representan su posición se asocian con la tabla de la siguiente forma:

- El eje vertical se ve representado por el primer valor, siendo 0 la posición más alta y añadiendo 1 cada vez que bajamos.
- El eje horizontal se ve representado por el segundo valor, siendo 0 la posición más a la izquierda y añadiendo 1 cada vez que nos desplazamos a la derecha.

Teniendo eso en cuenta se modificarán los pares de números que determinan la posición de los productos en el siguiente código:

```
int[] actual = {2, 1}; //inicializado en la entrada
             lugar.put("actual", actual);

             int[] entrada = {2,1};
             lugar.put("Entrada",entrada);

             int[] salida = {2,1};
             lugar.put("Salida",salida);

             int[] pollo = {3,0};
             lugar.put("Pollo",pollo);

             int[] ternera = {3,1};
             lugar.put("Ternera",ternera);

             int[] zanahoria = {2,4};
             lugar.put("Zanahoria",zanahoria);

             int[] puerro = {2,3};
             lugar.put("Puerro",puerro);

             int[] leche = {3,1};
             lugar.put("Leche",leche);

             int[] yogurt = {3,1};
             lugar.put("Yogurt",yogurt);

             int[] manzana = {0,2};
             lugar.put("Manzana",manzana);

             int[] platano = {1,2};
             lugar.put("Platano",platano);

             int[] salmon = {2,2};
             lugar.put("Salmon",salmon);

             int[] sardinas = {2,2};
                    lugar.put("Sardinas",sardinas);
```

6. Mantenimiento

El mantenimiento necesario para dicha instalación es mínimo. Los únicos detalles a tener en cuenta son los siguientes:

- Duración de las baterías
- Modificación de los elementos de la tienda

6.1 Duración de las baterías

En caso de que la batería de una baliza se agote deberá ser sustituida inmediatamente. En caso de no poder reemplazar la batería deberá sustituirse la baliza por otra y actualizar el programa para que reconozca su dirección.

Si el patrón de radiación de la nueva baliza es suficientemente similar al de la anterior no hará falta realizar nuevas mediciones, pero de no ser así será necesario volver a calcular los vectores de radiación que se viesen influidos por el alcance de dicha baliza.

Lo mismo se realizará en caso de sustracción no autorizada de la baliza si esta no puede recuperarse.

6.2 Modificación de los elementos de la tienda

En caso de que los productos de la tienda hayan sido desplazados a otra posición se deberá modificar dicho valor en la clase "Posiciones".

Si la modificación implica una nueva distribución de estanterías, pasillos y otros elementos quizás sea necesario volver a realizar todas las mediciones para obtener los nuevos vectores de potencia asociados a las posiciones, así como haber modificado el mapa en consecuencia.